Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ6

_____В.В. Сюзев
«___» _____ 2011 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине «Параллельные компьютерные системы»

Автор: Можаров Г.П.
Факультет – ИУ (каф. ИУ6)
Кафедра ИУ6 – «Компьютерные системы и сети»

Москва
2011

СОДЕРЖАНИЕ

Введение

Целью лабораторного практикума по параллельным компьютерным системам является знакомство студентов с разработанными методами решения некоторых прикладных и математических задач по комбинаторике, используемых при расчёте основных характеристик параллельных компьютерных системах (КС). Кроме того одна из целей – привлечь студентов к тематике экстремальных задач и к комбинаторике как к предмету исследований.

Подбор тем, затронутых в «Методических указаниях к лабораторным работам … », представляет некоторые методы решения комбинаторных задач, применяемые при расчёте как структуры, так и процессов в параллельных КС. В «Методических указаниях к лабораторным работам …» делается акцент на конструктивный алгоритмический подход – наряду с обсуждаемыми комбинаторными проблемами, приводятся алгоритмы их решения, зачастую вместе с анализом их вычислительной сложности.

Основная цель лабораторных работ состоит в изложении и демонстрации некоторых методов решения комбинаторных задач, используемых при расчёте и проектировании параллельных КС (изложенными в лекциях).

Расширение областей применения теоретических комбинаторных результатов приводит к зарождению важного проблемного направления – комбинаторного моделирования. При этом выбор наиболее подходящей комбинаторной трактовки прикладных задач определяется конечными целями их решения. Широкая степень абстракции каждой комбинаторной модели позволяет с их помощью исследовать некоторый определённый круг процессов или явлений из различных областей знаний. Следовательно, объединение таких моделей в комплексы, чей состав будет определяться путём нахождения правил соответствия между ними, которые, в свою очередь, будут зависеть от задач, решаемых с помощью таких комплексов моделей, существенным образом расширит области их применения. Это приводит к образованию ещё одного проблемного направления – изучению соответствий между различными моделями. Основная цель, которая преследуется этим проблемным направлением: создание унифицированных комплексов комбинаторных моделей, пригодных для адекватного описания не только специализированных задач практики, но и для описания процессов и явлений, принадлежащих некоторому кругу предметных областей знаний.

Задача лабораторного практикума – ознакомить студентов с некоторыми алгоритмами, используемыми при проектировании параллельных КС.

В первой лабораторной работе рассматриваются разбиения множества и чисел.

.

Вторая лабораторная работа посвящена вычислению наибольшего паросочетания методом Хопкрофта-Карпа; третья – жадным алгоритмам для матричного матроида и матроида трансверсалей.

Порядок выполнения и защиты лабораторных работ. Для проведения лабораторной работы в классе ЦВМ студент должен:

• ознакомиться с теоретическими сведениями и алгоритмами, связанными с параллельными системами;

• изучить среду программирования, используемую для решения задач построения параллельных систем;

Во время выполнения лабораторной работы на ЦВМ необходимо:

• в соответствии с предлагаемым алгоритмом написать и отладить программу;

• для этой программы подготовить 2-3 тестовых примера, иллюстрирующих работу программы;

• обеспечить режим пошагового выполнения алгоритмов с выдачей на дисплей получаемых промежуточных результатов;

• во всех случаях, когда результатом работы программы являются графы или временные диаграммы, необходимо обеспечить их выдачу на дисплей, используя графический метод доступа.

Самостоятельная подготовка студентов к каждой лабораторной работе требует около четырёх академических часов. После выполнения очередной лабораторной работы необходимо получить у преподавателя отметку о выполнении. Отчёт в соответствии с установленной формой представляется на очередном занятии. Отсутствие отчёта может служить причиной недопуска студента к следующей лабораторной работе.

Залогом успешного выполнения лабораторных работ является самостоятельная подготовка студента. Для облегчения этой подготовки в описании каждой работы есть теоретическая часть, в которой приводятся основные понятия, используемые в данной лабораторной работе, а также требуемые алгоритмы. Имеются контрольные вопросы, с помощью которых можно оценить степень готовности студента к выполнению лабораторной работы.

Работа считается выполненной и зачтённой, если она была защищена. Защита лабораторной работы заключается в демонстрации работы программы на ЦВМ, ответах на вопросы, связанные с её теоретическими аспектами и получаемыми численными характеристиками.

Лабораторная работа № 1

Генерирование разбиений множества и чисел

Цель работы – ознакомление с одним из разделов комбинаторики – разбиения *n*-элементного множества и чисел.

Теоретическая часть

Под *разбиением* *n*-элементного множества X на k блоков будем понимать произвольное семейство $\pi = \{B_1, \ldots, B_k\}$, такое что $B_1 \cup \ldots \cup B_k = X$, $B_i \cap B_j = \varnothing$ для $1 \le i < j \le k$ и $B_i \ne \varnothing$ для $1 \le i \le k$. Подмножества $B_1, \ldots, B_k$ будем называть *блоками* семейства π. Множество всех разбиений множества X на k блоков будем обозначать $\Pi_k(X)$, а множество всех разбиений через $\Pi(X)$. Очевидно, что $\Pi(X) = \Pi_1(X) \cup \ldots \cup \Pi_n(X)$ (более того, $\{\Pi_1(X), \ldots, \Pi_n(X)\}$ является разбиением множества $\Pi(X)$).

С разбиением связано понятие отношения эквивалентности. Каждому разбиению π мы можем поставить в соответствие отношение эквивалентности

$$E(\pi) = \bigcup_{B \in \pi} (B \times B) \tag{1.1}$$

(элементы $x, y \in X$ находятся в отношении $E(\pi)$ тогда и только тогда, когда они принадлежат одному и тому же блоку разбиения π). И наоборот, каждому отношению эквивалентности E на множестве X мы можем поставить в соответствие разбиение

$$X, \quad E = \{x, \quad E : x \in X\}, \tag{1.2}$$

где $x, \quad E$ обозначает *класс эквивалентности* элемента x, т. е. множество всех элементов, находящихся в отношении E к элементу x:

$$x, \quad E = \{y \in X : xEy\}. \tag{1.3}$$

Нетрудно заметить, что формулы (1.1) и (1.2) определяют взаимно однозначное соответствие между разбиениями и отношениями эквивалентности на множестве X.

Если $\pi, \sigma \in \Pi(X)$ и каждый блок $B \in \sigma$ является суммой некоторого числа блоков разбиения π, то будем говорить, что π есть *измельчение* разбиения σ, и будем писать $\pi \le \sigma$. Например:

$$\{\{1\}, \{2, 5\}, \{4, 6\}, \{3\}\} < \{\{1, 2, 3, 5\}, \{4, 6\}\}.$$

Легко проверить, что $\pi \le \sigma$ тогда и только тогда, когда для отношений, эквивалентности, соответствующих данным разбиениям, выполняется соотношение $E(\pi) \subseteq E(\sigma)$. Очевидно, что определённое таким образом отношение $\le$ является частичным упорядочением на множестве $\Pi(X)$. Некоторые свойства множества $\Pi(X)$, упорядоченного с помощью отношения измельчения, напоминают аналогичные свойства множества $\mathsf{P}(X)$, упорядоченного на основе включения. Остановимся сейчас на одном из таких свойств.

Частично упорядоченное множество $\langle A, \le \rangle$ будем называть *решёткой,* если для произвольных элементов $x, y \in A$ существуют такие элементы $a, b \in A$, что

(*а*) $a \le x, a \le y$ и для произвольного элемента c, такого что $c \le x, c \le y$, имеем $c \le a$,

(*б*) $b \ge x, b \ge y$ и для произвольного элемента c, такого что $c \ge x, c \ge y$, имеем $c \ge b$.

Такие элементы, если они существуют, однозначно определяются через x и y. Действительно, для произвольных элементов a_1, a_2, отвечающих условию (*а*), имеем $a_2 \le a_1$ и $a_1 \le a_2$, т. е. $a_1 = a_2$; аналогично доказывается (*б*). Элемент a будем называть *нижней границей* элементов x, y и обозначать $x \wedge y$, а элемент b будем называть *верхней границей* и обозначать $x \vee y$.

Примером решётки является $\langle \mathsf{P}(X), \subseteq \rangle$. Очевидно, что в этом случае имеем $A \wedge B = A \cap B$ и $A \vee B = A \cup B$.

Теорема 1.1. Множество $\Pi(X)$, упорядоченное на основе отношения измельчения $\le$, образует решётку, причём

$$\pi \wedge \sigma = \{A \cap B : (A \in \pi) \wedge (B \in \sigma) \wedge (A \cap B \ne \varnothing)\}, \tag{1.4}$$

т.е.

$$E(\pi \wedge \sigma) = E(\pi) \cap E(\sigma), \tag{1.5}$$

а разбиение $\pi \vee \sigma$ определяется следующим образом:

$\langle x, y \rangle \in E(\pi \vee \sigma) \Leftrightarrow$ существует $k \ge 1$ и такая последовательность $x_1, \ldots, x_k$, что $x = x_1$,

$$y = x_k \text{ и } \langle x_i, x_{i+1} \rangle \in E(\pi) \text{ или} \tag{1.6}$$

$$\langle x_i, x_{i+1} \rangle \in E(\sigma) \text{ для } i = 1, 2, \ldots, k-1.$$

Доказательство. Отметим, прежде всего, что простым следствием определения отношения эквивалентности является тот факт, что пересечение двух отношений эквивалентности есть отношение эквивалентности.

Таким образом, $E(\pi)\bigcap E(\sigma)$ есть отношение эквивалентности, которое в свою очередь однозначно определяет некоторое разбиение α, такое что $E(\pi)\bigcap E(\sigma)=E(\alpha)$ (ср. (1.2)). Очевидно, что $E(\alpha)\subseteq E(\pi)$, $E(\alpha)\subseteq E(\sigma)$, а, следовательно, $\alpha\le\pi$, $\alpha\le\sigma$. Более того, для каждого разбиения α', такого что $\alpha'\le\pi$, $\alpha'\le\sigma$, имеем $E(\alpha')\subseteq E(\pi)$, $E(\alpha')\subseteq E(\sigma)$ и в результате $E(\alpha')\subseteq E(\pi)\bigcap E(\sigma)=E(\alpha)$, т. е. $\alpha'\le\alpha$. Следовательно, $\alpha=\pi\wedge\sigma$, что доказывает формулу (1.5). Отметим, что $x,\ y\in E(\pi\wedge\sigma)=E(\pi)\bigcap E(\sigma)$ тогда и только тогда, когда x и y принадлежат к одному и тому же блоку разбиения π и к одному блоку разбиения σ, т.е. $x,\ y\in A\bigcap B$, где $A\in\pi,\ B\in\sigma$. Последнее и доказывает формулу (1.4).

Предположим теперь, что R – бинарное отношение, определённое правой частью формулы (1.6). Нетрудно проверить, что R – отношение эквивалентности. Действительно, xRx (достаточно принять $k = 1$), xRy влечёт за собой yRx (достаточно рассмотреть последовательность $x_k, x_{k-1}, ..., x_1$) и xRy, yRz влекут за собой xRz (достаточно рассмотреть конкатенацию соответствующих последовательностей). Отношение R однозначно определяет разбиение β, такое что $E(\beta) = R$. Очевидно, что $E(\pi)\subseteq R = E(\beta)$ и $E(\sigma)\subseteq R = E(\beta)$, т. е. $\pi\le\beta$ и $\sigma\le\beta$. Предположим теперь, что β' – произвольное разбиение, такое что $\pi\le\beta'$, $\sigma\le\beta'$, и пусть $(x, y)\in E(\beta)$. Тогда существует последовательность $x = x_1,\ x_2,\ \ldots,\ x_k = y$, удовлетворяющая правой части формулы (1.6), и, следовательно, $\langle x_i,\ x_{k+1}\rangle\in E(\pi)\bigcup E(\sigma)\subseteq E(\beta'),\ 1\le i\le k$. Вследствие транзитивности отношения $E(\beta')$ имеем $\langle x,\ y\rangle\in E(\beta')$. Таким образом, мы доказали, что $\beta\le\beta'$, а следовательно, и $\beta=\pi\vee\sigma$. ■

Числа Стирлинга второго и первого рода. *Число Стирлинга второго рода* $S(n,\ k)$ определяется как число разбиений n-элементного множества на k блоков:

$$S(n,\ k)=\left|\prod\nolimits_k(X)\right|, \tag{1.7}$$

где $|X|=n$.

Например, $S(4,\ 2)=7$, так как существуют в точности 7 разбиений множества {1, ..., 4} на два блока:

{{1, 2, 3}, {4}}

{{1, 2, 4}, {3}}

{{1, 3, 4}, {2}}

{{1, 2}, {3, 4}}

{{1, 3}, {2, 4}}

{{1, 4}, {2, 3}}

{{1}, {2, 3, 4}}

Очевидно, $S(n,\ k)=0$ для $k>n$. Примем также $S(0,\ 0)=1$, так как пустое семейство блоков является в соответствии с определением разбиением пустого множества. С числами Стирлинга второго порядка связано, как и с биномиальными коэффициентами, много любопытных тождеств.

Докажем сначала тождество, связанное с треугольником Паскаля:

$$S\left(n,\ k\right) = S\left(n-1,\ k-1\right) + kS\left(n-1,\ k\right) \text{ для } 0 < k < n, \tag{1.8}$$

$$S\left(n,\ n\right) = 1 \text{ для } n \geq 0, \tag{1.9}$$

$$S\left(n,\ 0\right) = 0 \text{ для } n > 0. \tag{1.10}$$

Формулы (1.9) и (1.10) очевидны. Для доказательства формулы (1.8) рассмотрим множество всех разбиений множества $\left\{1,\ \ldots,\ n\right\}$ на k блоков. Это множество распадается на два различных класса: тех разбиений, которые содержат одноэлементный блок $\left\{n\right\}$, и тех разбиений, для которых n является элементом большего (по крайней мере, двухэлементного) блока. Мощность первого класса равна $S\left(n-1,\ k-1\right)$, т.е. такова, каково число разбиений множества $\{1, ..., n-1\}$ на $k-1$ блоков. Мощность другого класса составляет $kS\left(n-1,\ k\right)$, так как каждому разбиению множества $\left\{1,\ \ldots,\ n-1\right\}$ на k блоков соответствует в этом классе в точности k разбиений, образованных добавлением элемента n поочерёдно к каждому блоку.

Формулы (1.8), (1.9) и (1.10) позволяют легко вычислять значения $S\left(n,\ k\right)$ даже для больших значений n и k. В табл. 1.1 представлены числа $S\left(n,\ k\right)$ для $0 \leq n,\ k \leq 10$.

Отметим, что эту таблицу можно трактовать как «треугольник Стирлинга», в котором каждое значение, кроме крайних, равных единице, можно получить как сумму чисел, находящихся над ним, а именно числа, расположенного в точности над ним и умноженного на k, и числа над ним с левой стороны.

Таблица 1.1 – Числа Стирлинга второго ряда

n \ k	0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	0	0	0	0	0	0	0	o
1	0	1	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0	0
3	0	1	3	1	0	0	0	0	0	0	0
4	0	1	7	6	1	0	0	0	0	0	0
5	0	1	15	25	10	1	0	0	0	0	0
6	0	1	31	90	65	15	1	0	0	0	0
7	0	1	63	301	350	140	21	1	0	0	0
8	0	1	127	966	1701	1050	266	28	1	0	0
9	0	1	255	3025	7770	6 951	2 646	462	36	1	0
10	0	1	511	9330	34105	42 525	22 827	5 880	750	45	1

Вот пример другой рекуррентной зависимости, связанной с числами Стерлинга второго рода:

$$S\left(n,\ k\right) = \sum_{i=k-1}^{n-1} \binom{n-1}{i} S\left(i,\ k-1\right), \text{ для } k \geq 2. \tag{1.11}$$

Для доказательства тождества рассмотрим множество всех разбиений $S\left(n,\ k\right)$ множества $X = \{1,..., n\}$. Это множество распадается на различные классы, соответствующие разным подмножествам множества X, которые являются блоками, содержащими элемент n.

Отметим, что для каждого 6-элементного подмножества $B \subseteq X$, содержащего элемент *n*, существует в точности $S\left(n-b,\ k-1\right)$ разбиений множества *X* на *k* блоков, содержащих *B* в качестве блока. Действительно, каждое такое разбиение однозначно соответствует разбиению множества $X \setminus B$ на *k* – 1 блоков. *b*-элементное множество $B \subseteq X$, содержащее элемент *n*, можно выбрать $\binom{n-1}{b-1}$ способами; таким образом,

$$S\left(n,\ k\right) = \sum_{b=1}^{n-(k-1)} \binom{n-1}{b-1} S\left(n-b,\ k-1\right) = \sum_{b=1}^{n-(k-1)} \binom{n-1}{n-1} S\left(n-b,\ k-1\right) = \sum_{i=k-1}^{n-1} \binom{n-1}{i} S\left(i,\ k-1\right).$$

Число Белла B_n определяется как число всех разбиений *n*-элементного множества

$$B_n = \left|\prod\left(X\right)\right|, \tag{1.12}$$

где $\left|X\right| = n$.

Другими словами,

$$B_n = \sum_{k=0}^{n} S\left(n,\ k\right). \tag{1.13}$$

Докажем теперь следующую простую рекуррентную зависимость, связанную с числами Белла:

$$B_{n+1} = \sum_{i=0}^{n} \binom{n}{i} B_i \tag{1.14}$$

(принимаем $B_0 = 1$). Доказательство проводится аналогично доказательству тождества (1.11). Множество всех разбиений множества *X* = {1,..., *n* +1} можно разбить на различные классы в зависимости от блока *B*, содержащего элемент *n* + 1, или – что равнозначно – в зависимости от множества $X \setminus B$. Для каждого множества $X \setminus B$\{1,..., *n*\} существует в точности $\left|\prod\left(X \setminus B\right)\right| = B_{|X \setminus B|}$ разбиений множества *X*, содержащих *B* в качестве блока. Группируя наши классы в зависимости от мощности множества $X \setminus B$, получаем формулу (1.14).

Числа B_n для $0 \le n \le 20$ представлены в табл. 1.2.

Таблица 1.2 – Числа Белла B_n

n	B_n
0	1
1	1
2	2
3	5
4	15
5	52
6	203
7	877
8	4 140
9	21 147

10	115 975
11	678 570
12	4 213 597
13	27 644 437
14	190 899 322
15	1 382 958 545
16	10 480 142 147
17	82 864 869 804
18	682 076 806 159
19	5 832 742 205 057
20	51724 158 235 372

Существует строгая зависимость между числами $S(n,\ k)$ и числом всех функций из *n*-элементного множества на *k*-элементное множество, т. е. функций $f: X \to Y$, $f(X) = Y$ для $|X| = n$, $|Y| = k$. Каждой такой функции f можно поставить в соответствие следующее разбиение множества *X* на *k* блоков:

$$N(f) = \left\{ f^{-1}(y) : y \in Y \right\}, \tag{1.15}$$

называемое *ядром* функции f (условие $f(X) = Y$ даёт гарантию того, что подмножества $f^{-1}(y)$ непустые). С другой стороны, нетрудно заметить, что каждому разбиению $\pi \in \prod_k (X)$ соответствует в точности *k*! функций из *X* на *Y*, таких что $N(f) = \pi$. Каждая такая функция взаимно однозначно соответствует соотнесению блоков разбиения я элементам множества *Y*. Обозначая через $s_{n,k}$ число функций из *X* на *Y*, получаем, следовательно,

$$s_{n,k} = k!\,S(n,\ k). \tag{1.16}$$

Пользуясь этой формулой, мы можем доказать ещё одно свойство чисел Стирлинга второго рода, которое касается связи между многочленами x^k и многочленами

$$[x]_k = x(x-1)\ \ldots\ (x-k+1). \tag{1.17}$$

Произвольный многочлен $P(x)$ от неизвестного *x* степени *n* мы можем однозначно представить как $P(x) = \sum_{k=0}^{n} a_k [x]_k$. Это частный случай того очевидного факта, что существует однозначное разложение $P(x) = \sum_{k=0}^{n} a_k p_k(x)$ для произвольной последовательности многочленов $p_0(x)$, $p_1(x)$, ..., такой что $p_k(x)$ есть многочлен степени *k* для каждого $k \geq 0$. Другими словами, каждая такая последовательность образует базис в линейном про-

странстве многочленов. Оказывается, что числа Стирлинга второго рода в точности равны коэффициентам перехода от базиса 1, x, x^2, ... к базису 1, $[x]_1$, $[x]_2$, ...

Теорема 1.2. Для каждого $n \geq 0$

$$x^n = \sum_{k=0}^{n} S(n,\ k)[x]_k . \tag{1.18}$$

Доказательство. Предположим сначала, что x – неотрицательное целое число. Подсчитаем двумя способами число всех функций $f : A \to B$, где $|A| = n$, $|B| = x$. С одной стороны, оно равно x^n (см. теорему 1.1). С другой стороны, множество таких функций f мы можем классифицировать относительно множества $f(A)$. Очевидно, что каждая функция f является отображением множества A на множество $f(A)$, таким образом, для произвольного подмножества $Y \subseteq B$, где $|Y| = k$, число всех функций $f : A \to B$, таких что $f(A) = Y$, равно $s_{n,k}$, т. е. в соответствии с формулой (1.41) $k!S(n,\ k)$. Учитывая тот факт, что подмножество Y мощности k можно выбрать $\binom{n}{k}$ способами, получаем в конечном итоге

$$S(n,\ k) = \sum_{i=k-1}^{n-1} \binom{n-1}{i} S(i,\ k-1), \tag{1.19}$$

(верхний индекс суммирования можно заменить с x на n, так как $S(n,\ k) = 0$ для $k > n$ и $[x]_k = 0$ для $k < x$). Поскольку равенство многочленов выполняется для произвольного целого числа $x \geq 0$, эти многочлены тождественно равны (так как их разность либо является тождественно равной нулю, либо имеет бесконечное число нулей). ■

Числа Стирлинга первого рода $s(n,\ k)$ определяются как коэффициенты при последовательных степенях переменной x в многочлене $[x]_k$:

$$[x]_n = \sum_{k=0}^{n} s(n,\ k)x^k . \tag{1.20}$$

Другими словами, числа $s(n,\ k)$ играют обратную роль в отношении к числам $S(n,\ k)$ – позволяют перейти от базиса 1, $[x]_1$, $[x]_2$, ..., к базису 1, x, x^2, ... Очевидно, что $s(n,\ k) = 0$ для $k > n$. Числа $s(n,\ k)$ удобно вычислять, пользуясь следующими рекуррентными зависимостями:

$$s(n,\ k) = s(n-1,\ k-1) - (n-1)s(n-1,\ k) \text{ для } 0 < k < n, \tag{1.21}$$

$$s(n,\ n) = 1 \text{ для } n \geq 0, \tag{1.22}$$

$$s(n,\ 0) = 0 \text{ для } n > 0. \tag{1.23}$$

Формулы (1.22) и (1.23) очевидны, формулу (1.21) получаем, сравнивая коэффициенты при x^k с обеих частей равенства

$$[x]_n = [x]_{n-1}(x - n + 1). \tag{1.24}$$

То есть имеем

$$\sum_{k=0}^{n} s\left(n,\ k\right)x^{k} = \left(x-n+1\right)\sum_{k=0}^{n-1} s\left(n-1,\ k\right)x^{k} = \sum_{k=0}^{n-1} s\left(n-1,\ k\right)x^{k+1} - \left(n-1\right)\sum_{k=0}^{n-1} s\left(n,\ k\right)x^{k} =$$

$$= \sum_{k=1}^{n-1}\left(s\left(n-1,\ k-1\right)-\left(n-1\right)s\left(n-1,\ k\right)\right)x^{k} + s\left(n-1,\ n-1\right)x^{n} - \left(n-1\right)s\left(n-1,\ 0\right).$$

В табл. 1.3 представлены числа $s\left(n,\ k\right)$ для $0 \le n,\ k \le 10$.

Таблица 1.3 – Числа Стирлинга первого рода

n \ k	0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
2	0	–1	1	0	0	0	0	0	0	0	0
3	0	2	–3	1	0	0	0	0	0	0	0
4	0	–6	11	–6	1	0	0	0	0	0	0
5	0	24	–50	35	–10	1	0	0	0	0	0
6	0	–120	274	–225	85	–15	1	0	0	0	0
7	0	720	–1 764	1 624	–735	175	–21	1	0	0	0
8	0	–5 040	13 068	–13 132	6769	–1 960	322	–28	1	0	0
9	0	40 320	–109 584	118 124	–67 284	22 449	–4 536	546	–36	1	0
10	0	–362 880	1 026 576	–1 172 700	723 680	–269325	63 273	-9 450	870	–45	0

Генерирование разбиений множества. Опишем теперь алгоритм генерирования всех разбиений множества. Идею этого алгоритма легче всего объяснить, сформулировав его в рекуррентной форме. Отметим сначала, что каждое разбиение π множества $\{1, ..., n\}$ однозначно определяет разбиение π_{n-1} множества $\{1, ..., n-1\}$, возникшее из я после удаления элемента n из соответствующего блока (и удаления образовавшегося пустого блока, если элемент n образовывал одноэлементный блок). Напротив, если дано разбиение $\sigma = \{B_1, ..., B_k)$ множества $\{1, ..., n-1\}$, легко найти все разбиения я множества $\{1, ..., n\}$, такие что $\pi_{n-1} = \sigma$, т.е, следующие разбиения:

$$\begin{gathered} B_1 \cup \{n\},\ B_2,\ ...,\ B_k, \\ B_1,\ B_2 \cup \{n\},\ ...,\ B_k, \\ \dots\dots\dots\dots\dots\dots \\ B_1,\ B_2,\ ...,\ B_k \cup \{n\}, \\ B_1,\ B_2,\ ...,\ B_k\ \{n\}. \end{gathered} \tag{1.25}$$

Это подсказывает следующий простой рекуррентный метод генерирования всех разбиений: если нам дан список L_{n-1} всех разбиений множества $\{1, ..., n-1\}$, то список L_n всех разбиений множества $\{1, ..., n\}$ будем создавать, заменяя каждое разбиение σ в списке L_{n-1} на соответствующую ему последовательность (1.25).

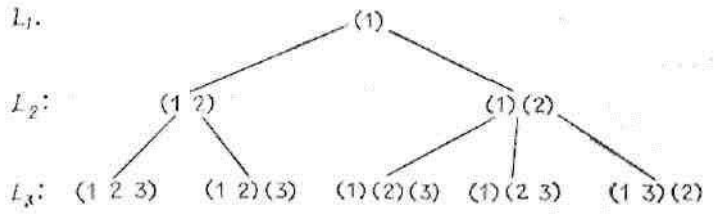


Рисунок 1.1 – Построение списков L_1 L_2, L_3.

Отметим, что при этом мы можем гарантировать, что разбиения, идущие друг за другом, будут мало отличаться друг от друга, точнее говоря, мы можем принять, что каждое следующее разбиение в списке образуется из предыдущего посредством удаления некоторого элемента из некоторого блока (это может повлечь за собой удаление одноэлементного блока) и добавления его в другой блок либо создания из него одноэлементного блока. Действительно, последовательные разбиения последовательности (1.25) отвечают этому условию. Если обратить порядок последовательности (1.25) для каждого второго разбиения списка L_{n-1}, то элемент n будет двигаться попеременно вперёд и назад, и разбиения «на стыке» последовательностей, образованных из соседних разбиений списка L_{n-1}, будут мало отличаться друг от друга (при условии, что соседние разбиения списка L_{n-1} мало отличаются один от другого). На рис. 1.1 показано построение списка L_n для n = 1, 2, 3 (разбиения представлены в несколько упрощённой записи, например (1 2) (3) означает {{1 2}, {3}}).

Дадим сейчас нерекуррентную реализацию этого алгоритма. Разбиение множества {1, ..., n} мы будем представлять с помощью последовательности блоков, упорядоченной по возрастанию самого маленького элемента в блоке. Этот наименьший элемент блока мы будем называть номером блока. Отметим, что номера соседних блоков, вообще говоря, не являются соседними натуральными числами. В этом алгоритме мы будем использовать переменные *ПРЕД*[i], *СЛЕД*[i], $1 \le i \le n$, содержащие соответственно номер предыдущего и номер следующего блока для блока с номером i (*СЛЕД*[i] = 0, если блок с номером i является последним блоком разбиения). Для каждого элемента i, $1 \le i \le n$, номер блока, содержащего элемент i, будет храниться в переменной *БЛОК*[i], направление, в котором «движется» элемент i, будет закодировано в булевской переменной *ВПЕР*[i] (*ВПЕР* [i] = **true,** если i движется вперёд; здесь можно заметить некоторое подобие алгоритму 1.12).

Алгоритм 1.1. *Генерирование всех разбиений множества* 1, ..., n.

Данные: n.

Результат: Последовательность всех разбиений множества {1, ..., n}, в которой каждое следующее разбиение образуется из предыдущего путём перенесения единственного элемента в другой блок.

1 **begin**

```
2    for i: = 1 to n do (* поместить i в первый блок *)
3        begin БЛОК [i]: = 1; ВПЕР [i]: = истина;
4        end;
5     СЛЕД [1]: = 0;
6     выписать разбиение;
7     j: =n; (*j = активный элемент *)
8     while j > 1  do
9        begin k: = БЛОК [j];
10          if  ВПЕР [j] then (* j движется вперёд *)
11            begin
12                if СЛЕД [k] = 0 then (* k есть последний блок *)
13                begin СЛЕД [k]: = j; ПРЕД [j]: = k; СЛЕД [j]: = 0
14                end;
15     if СЛЕД [k] > j then (* j  образует новый блок *)
16                begin ПРЕД [j]: = k; СЛЕД [j]:=СЛЕД [k]
17                              ПРЕД [СЛЕД [j]]: = j; СЛЕД [k]: = j
18                 end;
19                 БЛОК [j]: = СЛЕД [k]
20              end
21        else (* j  движется назад *)
22              begin БЛОК [j]: = ПРЕД [k];
23                if k = j then (* jобразует одноэлементный блок *)
24                   if СЛЕД [k] = 0 then СЛЕД [ПРЕД [k]]: = 0
25                   else begin СЛЕД [ПРЕД [k]]: = СЛЕД [k];
26                         ПРЕД [СЛЕД [k]]: = ПРЕД [k]
27                         end
28              end;
29              выписать разбиение;
30              j: = n;
31              while (j > 1) end
32              ((ВПЕР [j] and (БЛОК [j] = j)) or (not ВПЕР [j] and (БЛОК [j]=1))) do
33                begin ВПЕР [j] = not ВПЕР [j]: j: = j– 1
34                end
35        end
36   end
```

Этот алгоритм строит сначала разбиение **{{**1, ..., n}} (цикл 2) – отметим, что это первое разбиение в списке L_n, созданном при помощи описанного нами рекуррентного метода. Задача основного цикла 8 – перемещение «активного» элемента j в соседний блок – предыдущий или последующий (в последнем случае может возникнуть необходимость создания нового блока вида $\{j\}$), а затем определение активного элемента во вновь образованном разбиении. Из описанного рекуррентного построения следует, что данный элемент перемещается только тогда, когда все элементы, большие его, достигают своего крайнего левого или правого положения; точнее, активный элемент j^* является таким наименьшим элементом, что для каждого большего элемента j выполняется одно из двух следующих условий:

(1) *ВПЕР*[j] **and** (*БЛОК* [j] = j), т.е. элемент движется вперёд и достигает своего крайнего правого положения (очевидно, j не может быть элементом блока с наименьшим элементом, большим j).

(2) **not** *ВПЕР*[j] **and** (*БЛОК* [j] = 1), т.е. элемент j движется назад и достигает своего крайнего левого положения (в первом блоке),

Этот принцип реализуется в строках 30-34. Заодно меняется направление движения элементов $j > j^*$. Дополнительным условием цикла 31 является $j > 1$, так как из самого представления разбиения следует, что $j = 1$ не может быть активным элементом (очевидно, что элемент 1 всегда является элементом блока с номером 1). Если каждый из элементов $j > 1$ отвечает условию (1) или (2), то легко убедиться, что уже порождены все разбиения. В таком случае на выходе цикла 31 имеем $j = 1$ и следует выход из основного цикла 8, т. е. имеем окончание работы алгоритма. Из рекуррентности алгоритма вытекает также, что активным элементом для первого разбиения списка L_n, т. е. для $\{\{1, ..., n\}\}$, является элемент n. Такое же значение приписывается переменной j перед входом в цикл 8 (строка 7).

Проанализируем теперь процесс переноса активного элемента (строки 9-28). Сначала отыскивается номер блока, содержащего активный элемент; пусть это будет k. Если этот элемент движется вперёд, достаточно перенести его в блок с номером *СЛЕД* [k] (см. строку 19), а в двух остальных случаях переменную *СЛЕД* [k] нужно сначала модифицировать. Первый случай имеет место, когда *СЛЕД* [k] = 0, т.е. когда k есть номер последнего блока разбиения. Тогда j образует одноэлементный блок; при этом достаточно принять *СЛЕД* [k] := j и соответственно изменить значения переменных *СЛЕД* [j] и *ПРЕД* [j] (см. строку 13). Второй случай имеет место, когда *СЛЕД* [k] > k, он рассматривается аналогично. Условие *СЛЕД* [k] > j означает, что все блоки справа от блока с номером k содержат элементы, большие j (все эти элементы занимают свои крайние правые позиции, в противном случае j не был бы активным элементом). Из рекуррентности алгоритма легко вытекает, что в этом случае нужно создать одноэлементный блок, содержащий j. Это выполняется в строках 16-17 (единственная разница с первым случаем состоит в том, что в данном случае вновь созданный блок не является последним блоком разбиения).

В ситуации, когда элемент j движется назад (см. строку 21), достаточно поместить его в предыдущий блок (см. строку 22) и выполнить соответствующее изменение значений переменных *СЛЕД* и *ПРЕД*, если j создавал одноэлементный блок – это имеет место в точности тогда, когда *БЛОК* [j] = $k = j$, так как каждый элемент $m > j$ блока с номером j был бы выбран активным элементом в цикле 31.

На рис. 1.2 представлены все разбиения множества $\{1, ..., 4\}$, порождённые алгоритмом. Можно показать, что среднее число шагов, необходимых для построения каждого следующего разбиения, ограничено постоянной, не зависящей от n.

$(1\ 2\ 3\ 4)$

$(1\ 2\ 3)(4)$

$(1\ 2)(3)(4)$

$(1\ 2)(3\ 4)$

$(1\ 2\ 4)(3)$

$(1\ 4)(2)(3)$

$(1)(2\ 4)(3)$

$(1)(2)(3\ 4)$

$(1)(2)(3)(4)$

$(1)(2\ 3)(4)$

$(1)(2\ 3\ 4)$

$(1\ 4)(2\ 3)$

$(1\ 3\ 4)(2)$

$(1\ 3)(2\ 4)$

$(1\ 3)(2)(4)$

Рисунок 1.2 – Последовательность разбиений множества {1, 2, 3, 4}, порождённая алгоритмом 1.1.

Разбиения чисел. Займёмся теперь следующей задачей: сколькими способами можно записать данное натуральное число *n* в виде суммы

$$n = b_1 + \ \dots \ + b_k, \tag{1.26}$$

где $k,\ b_1,\ \dots,\ b_k > 0$. При этом будем считать суммы эквивалентными, если они отличаются только порядком слагаемых. Класс эквивалентных сумм вида (1.26) мы можем однозначно представить последовательностями $a_1,\ \dots,\ a_k$, где $a_1 \geq \ \dots \ \geq a_k$ и числа $a_1,\ \dots,\ a_k$ являются числами $b_1,\ \dots,\ b_k$, упорядоченными по невозрастанию (аналогичным образом мы представляли подмножества множества возрастающими последовательностями и подмножества множества с повторениями неубывающими последовательностями). Каждую такую последовательность $a_1,\ \dots,\ a_k$ будем называть *разбиением числа n* на *k* слагаемых. Число разбиений числа *n* на *k* слагаемых будем обозначать $P\left(n,\ k\right)$, а число всех разбиений числа *n* (на произвольное число слагаемых) – через $P\left(n\right)$. Очевидно, что

$$P\left(n\right) = \sum_{k=1}^{n} P\left(n,\ k\right), \quad n > 0 \tag{1.27}$$

(принимаем $P\left(0,\ 0\right) = P\left(0\right) = 1\right)$. Например, $P\left(7\right) = 15$, а все разбиения числа 7 представлены на рис. 1.3.

```
7
6 1
5 2
5 1 1
4 3
4 2 1
4 1 1 1
3 3 1
3 2 2
3 2 1 1
3 1 1 1 1
2 2 2 1
2 2 1 1 1
2 1 1 1 1 1
1 1 1 1 1 1 1
```

Рисунок 1.3 – Последовательность разбиений числа 7, построенная с помощью алгоритма 1.2

Используя очень простой способ представления разбиения числа, называемый *диаграммой Ферре*, можно доказать много любопытных свойств чисел $P(n)$. Диаграмма Ферре для разбиения $n = a_1 + \ldots + a_k$ состоит из k строк, соответствующих слагаемым разбиения, причём i-я строка содержит последовательность из a_i точек (рис. 1.4).

```
• • • • • •          • • • •
• • • •              • • • •
• • • •              • • •
• •                  • • •
                     •
                     •
16 = 6 + 4 + 4 + 2   16 = 4 + 4 + 3 + 3 + 1 + 1
```

Рисунок 1.4 – Диаграмма Ферре для разбиения числа и сопряжённого ему разбиения

Каждому разбиению числа n однозначно соответствует *сопряжённое разбиение* этого числа, которое получается транспозицией диаграммы Ферре (перемена ролями строк и столбцов). Легко заметить, что транспозиция диаграммы Ферре определяет взаимно однозначное соответствие между разбиениями числа n на k слагаемых и разбиениями числа n с наибольшим слагаемым, равным k.

Отметим этот факт:

Теорема 1.3. Число разбиений числа n на k слагаемых равно числу разбиений числа n с наибольшим слагаемым, равным k. ■

Докажем ещё одну теорему этого типа.

Теорема 1.4. Число разбиений числа n на попарно различимые слагаемые равно числу разбиений числа n на нечётные слагаемые.

Доказательство. Установим взаимно однозначное соответствие между разбиениями, о которых идёт речь в теореме.

Рассмотрим разбиение числа n на нечётные слагаемые $b_1, \ldots, b_p$, где слагаемое b_i появляется в разбиении r_i раз, $1 \le i \le p$. Пусть

$$r_i = 2^{q_1} + 2^{q_2} + \ \dots \ \left(q_1 > q_2 > \ \dots \ \right)$$

есть двоичное представление числа r_i. Произведём теперь замену r_i слагаемых b_i на попарно различимые слагаемые

$$b_i 2^{q_1},\ b_i 2^{q_2},\ \dots$$

(эта замена сохраняет сумму слагаемых разбиения). Повторяя эту операцию для каждого $i,\ 1 \le i \le p$, и упорядочивая слагаемые по невозрастанию, получаем в результате разбиения числа *n* на попарно различимые слагаемые. Это следует из того факта, что каждое натуральное число можно однозначно представить в виде произведения нечётного числа на степень двойки. В качестве примера проведём описанное преобразование для разбиения 26 = 7 + 5 + 5 + 3 + 3 + 1 + 1 + 1:

$$7 + 5 + 5 + 3 + 3 + 1 + 1 + 1 = 7\cdot 2^0 + 5\cdot 2^1 + 3\cdot 2^1 + 1(2^1 + 2^0) = 7 + 10 + 6 + 2 + 1 = 10 + 7 + 6 + 2 + 1.$$

Легко заметить, что можно выполнить обратное преобразование для произвольного разбиения на попарно различимые слагаемые, представляя каждое слагаемое как $p2^q$, где p нечётное, группируя затем слагаемые в зависимости от «нечётного множителя» p и заменяя каждую такую группу $p2^{q_1},\ p2^{q_2},\ \dots$ на $r = 2^{q_1} + 2^{q_2} + \ \dots$ слагаемых, равных p. Таким образом, описанное преобразование определяет взаимно однозначное соответствие между разбиениями на нечётные слагаемые и разбиениями на попарно различимые слагаемые. ■

Покажем теперь простой алгоритм генерирования всех разбиений числа. Он будет генерировать разбиения в порядке, обратном лексикографическому, т.е. разбиение

$$n = c_1 + \ \dots \ + c_l \tag{1.28}$$

будет порождено – необязательно непосредственно – после разбиения

$$n = a_1 + \ \dots \ + a_k \tag{1.29}$$

тогда и только тогда, когда существует индекс $p \le \min\left(k,\ l\right)$, такой что $c_p < a_p$ и $c_m = a_m$ для $1 \le m < p$ (ср. с рис. 1.3). Очевидно, что первым разбиением в этом порядке является разбиение, содержащее одно слагаемое, равное *n*, а последним – разбиение на *n* слагаемых, равных единице. Зададимся вопросом, как выглядит разбиение, непосредственно следующее за разбиением (1.29). Будем искать разбиение, которое имеет самое большое число начальных слагаемых, равных начальным слагаемым разбиения (1.29) – обозначим эти слагаемые $a_1,\ \dots,\ a_{t-1}$, – и оставшиеся слагаемые которого определяются разбиением, непосредственно идущим за разбиением $s = a_t + a_{t+1} + \ \dots \ + a_k$. Легко заметить, что эти условия однозначно определяют

$$t = \max\left\{i : a_i > 1\right\}.$$

Таким образом, задача сводится к нахождению разбиения, непосредственно идущего за разбиением

$$s = a_t + \underbrace{1 + \ \dots \ + 1}_{k-t \text{ раз}},\quad a_t > 1.$$

Нетрудно заметить, что такое разбиение имеет вид

$$s = \underbrace{l + \ \ldots \ + l}_{\lfloor s/l \rfloor \text{ раз}} + \left(s \bmod l \right),$$

где $l = a_t - 1$.

В алгоритме, который мы сейчас опишем, разбиение будет представлено переменными $S[1]$, ..., $S[d]$, содержащими попарно различимые слагаемые разбиения ($S[1] > \ \ldots > S[d]$), а также переменными $R[1]$, .., $R[d]$, где $R[i]$ содержит ин« формацию о том, сколько раз слагаемое $S[i]$ появляется в разбиении ($R[i] > 0$). Это позволяет находить каждое следующее разбиение за число шагов, ограниченное некоторой постоянной, не зависящей от n.

Алгоритм 1.2 (*Нахождение всех разбиений числа*).

Данные: n.

Результат: последовательность разбиений числа n в порядке, обратном легсикографическому.

```
begin
   S[1]: = n; R[1]: = 1; d: = 1; (* первое разбиение *)
   выписать разбиение;
   while S[1] > 1 do (* нахождение следующего разбиения *)
      begin sum: = 0;  (* sum = сумма  устранённых слагаемых *)
         if S[d] = l then (* удаление слагаемых, равных единице *)
            begin sum: = sum + R[d];  d: = d – 1
            end;
         sum: = sum + S[d]; R[d]: = R[d] – 1; l: = S[d] – 1;
          if R[d] > 0 then d: = d+ 1;
          S[d]: = l; R[d]: = sum div l;
         l: = sum mod l;
          if l ≠ 0 then (* добавить  последнее  слагаемое, равное l *)
             begin d: = d+1; S[d]: = l; R[d]: = 1
             end:
          выписать разбиение
       end
 end
```

Вопросы для самопроверки

1. Что понимается под разбиением n-элементного множества X на k блоков ?
2. Что понимается под классом эквивалентности элемента x ?
3. Что означает термин «решётка» ?
4. Как определяются числа Стирлинга первого и второго родов ?
5. Как определяется число Белла?
6. Какое разбиение получается в результате транспозиции диаграммы Ферре ?

Задание на лабораторную работу

1. По заданным **алгоритмам 1.1** и **1.2** написать и отладить программы для генерирования всех разбиений множества 1, ..., n и разбиений числа.

2. Продемонстрировать работающие программы преподавателю и получить отметку о ее выполнении.

3. Провести анализ полученного алгоритма.

4. Оформить отчёт о проделанной работе. Он должен включать в себя:

– цель работы;

– ответы на вопросы для самопроверки;

– схему обрабатывающего алгоритма и описание его работы;

– распечатки экранных форм, полученных в результате работы программы.

Лабораторная работа № 2

Вычисление наибольшего паросочетания методом Хопкрофта-Карпа

Цель работы – написать и отладить программу нахождения наибольшего паросочетания методом Хопкрофта-Карпа.

Теоретическая часть

Наибольшие паросочетания в двудольных графах. *Паросочетанием* в неориентированном графе $G = \langle V,\ E \rangle$ называется произвольное множество рёбер $M \subseteq E$, такое, что никакие два ребра из M не инцидентны одной вершине. Другими словами, M есть паросочетание, если для каждых двух рёбер $e_1,\ e_2 \in M$ имеем либо $e_1 = e_2$, либо $e_1 \cap e_2 = \varnothing$. Для каждого ребра $\{u,\ v\} \in M$ мы говорим, что M *сочетает* u с v, а каждую вершину, не принадлежащую ни одному ребру паросочетания, будем называть *свободной*. Особенно интересны задачи, связанные с паросочетаниями в *двудольных* графах, т.е. в неориентированных графах $G = \langle V,\ E \rangle$, таких, что множества их вершин можно разбить па непересекающиеся множества $V = X \cup Y,\ X \cap Y = \varnothing$, причём каждое ребро $e \in E$ имеет вид $e = \{x,\ y\}$, где $x \in X,\ y \in Y$. Двудольный граф будем обозначать $\langle X,\ Y,\ E \rangle$ (это обозначение предполагает фиксацию некоторого разбиения множества вершин на множества X и Y – оно не определяется однозначно через множества V è E).

В этом разделе мы займёмся задачей нахождения наибольшего паросочетания в заданном двудольном графе. Это классическая задача комбинаторики, известная также под названием «задачи о супружеских парах». Такое название связано со следующей интерпретацией. Пусть X è Y – соответственно множества юношей и девушек, и пусть $\{x,\ y\} \in E$ означает, что юноша *x* знаком с девушкой *y*. Тогда каждое паросочетание M соответствует возможному множеству супружеских пар, в котором каждая пара образована из юноши и девушки, знакомых между собой, причём каждый человек участвует не более чем в одной паре.

Оказывается, что задачу нахождения наибольшего паросочетания в двудольном графе можно простым способом свести к построению максимального потока в некоторой сети. Пусть $H = \langle X,\ Y,\ E \rangle$ – произвольный двудольный граф. Построим сеть $S(H)$ с источником s, стоком r $(s \neq t$ è $s,\ t \notin X \cup Y)$, множеством вершин

$$V^* = \{s,\ t\} \cup X \cup Y,$$

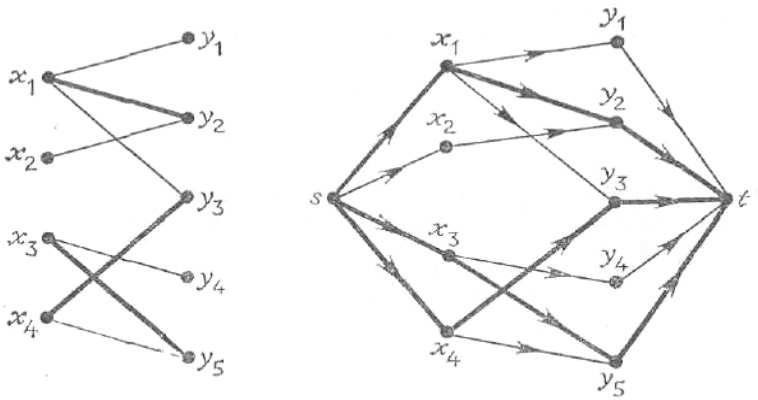
множеством дуг

$$E^* = \{\langle s,\ t\rangle : x \in X\} \cup \{\langle y,\ t\rangle : y \in Y\}$$
$$U\ \{\langle x,\ y\rangle : x \in X \wedge y \in Y \wedge \{x,\ y\} \in E\}$$

и пропускной способностью $c(u,\ v) = 1$ для каждой дуги $\langle u,\ v\rangle \in E^*$.

На рис. 2.1 показано построение сети $S(H)$ для некоторого двудольного графа.

Отметим, что в $S(H)$ существует максимальный нуль-единичный поток, т.е. максимальный поток f такой, что $f(e) = 0$ или $f(e) = 1$ для каждой дуги $e \in E^*$.



H *S*(*H*)

Рисунок 2.1 – Построение сети $S(H)$ для двудольного графа H и нуль-единичный поток, соответствующий паросочетанию $M = \{\{x_1,\ y_2\},\ \{x_3,\ y_5\},\ \{x_4,\ y_3\}\}$

Теорема 2.1. Существует взаимно однозначное соответствие между паросочетаниями в H и нуль-единичными потоками в $S(H)$, при котором паросочетанию $M = \{\{x_1,\ y_1\},\ \ldots,\ \{x_k,\ y_k\}\}$ мощности k $(x_i \in X,\ y_i \in Y$ äëÿ $1 \le i \le k)$ соответствует поток f_M величины k, определяемый следующим образом:

$$f_M(s,\ x_i) = f_M(x_i,\ y_i) = f_M(y_i,\ t) = 1,\quad 1 \le i \le k, \tag{2.1}$$

и $f_M(e) = 0$ для остальных дуг е сети $S(H)$; потоку f величины k соответствует паросочетание M_f, $= |M_f| = k$, определяемое следующим образом:

$$M_f = \{\{x,\ y\} : x \in X \wedge y \in Y \wedge f\{x,\ y\} = 1\}. \tag{2.2}$$

Доказательство. Если $M = \{\{x_1,\ y_1\},\ \ldots,\ \{x_k,\ y_k\}\}$ – паросочетание мощности k, то вершины $x_1,\ \ldots,\ x_k$, а также $y_1,\ \ldots,\ y_k$ попарно различны. Отсюда сразу же следует, что

$\mathrm{Div}_{f_M} = \mathrm{Div}_{f_M}(y_i) = 0,\ 1 \le i \le k$, т.е. функция f_M, определяемая формулой (4.1), является потоком из s â t в $S(H)$. Легко заметить также, что разным паросочетаниям M соответствуют разные потоки f_M. С другой стороны, если f – нуль-единичиый поток в $S(H)$, то количество потока, прибывающего в (а следовательно, и вытекающего из) каждую вершину $x \in X$, не превосходит единицы {единственная дуга, входящая в x, – это дуга $\langle s,\ x\rangle$ с пропускной способностью, равной 1). Отсюда следует, что в множестве M_f, определяемом (2.2), нет ребер вида $\{x,\ y_1\}$, $\{x,\ y_2\}$, $y_1 \ne y_2$. По аналогичным причинам в M_f нет ребер вида $\{x_1,\ y\}$, $\{x_2,\ y\}$, $x_1 \ne x_2$. Следовательно, M_f является паросочетанием в H. Нетрудно также отметить, что отображение $f \mapsto M_f$ является обратным к отображению $M \mapsto f_M$, т. е. $f_{M_f} = f$ и $M_{f_M} = M$. ■

Теорема 2.1 позволяет использовать произвольный алгоритм построения максимального потока (целочисленного) для определения максимального паросочетания. Так, например, используя алгоритм «правильного построения максимального потока в сети», мы находим наибольшее паросочетание за $O(n^3)$ шагов. Оказывается, однако, что особая форма сети $S(H)$ позволяет построить более эффективный алгоритм. Этот алгоритм, авторами которого являются Хопкрофт и Карп, использует общую схему Диница (метод Диница основан на построении некоторой вспомогательной бесконтурной сети (т.е. с графом, не содержащим контуров), структура которой точно отображает все кратчайшие увеличивающие цепи из *s* в *t* относительно фактического потока *f*), т.е. состоит из фаз, соответствующих увеличению потока вдоль кратчайших увеличивающихся цепей определённой длины. Особый вид сети позволяет, однако, как ограничить число фаз, так и построить более эффективный алгоритм построения псевдомаксимального потока в каждой фазе.

Отметим сначала, что каждая увеличивающая цепь в $S(H)$ будет нечётной длины и будет состоять после отбрасывания первой и последней дуг из последовательности попеременно чередующихся согласованных и несогласованных дуг (начинающейся и заканчивающейся согласованной дугой). Для данного паросочетания M мы будем называть *чередующейся цепью* (длины $l = 2k+1$ из X â Y) относительно M произвольное множество дуг $P \subseteq E$ вида

$$P = \{\{x_0,\ y_1\},\ \{y_1,\ x_1\},\ \{x_1,\ y_2\},\ \ldots,\ \{y_k,\ x_k\},\ \{x_k,\ y_{k+1}\}\}, \tag{2.3}$$

где $k > 0$, все вершины $x_0,\ \ldots,\ x_k,\ y_1,\ \ldots,\ y_{k+1}$ различны, x_0 – свободная вершина в X, y_{k+1} – свободная вершина в Y, каждая вторая дуга принадлежит M, т.е.

$$P \cap M = \{\{y_1,\ x_1\},\ \{y_2,\ x_2\},\ \ldots,\ \{y_k,\ x_k\}\}.$$

Очевидно, что чередующуюся цепь можно однозначно определить последовательностью $x_0,\ y_1,\ x_1,\ y_2,\ \ldots,\ y_k,\ x_k,\ y_{k+1}$. Разумеется, что при описанном в **теореме 2.1** соответствии между паросочетаниями в H и нуль-единичными потоками в сети $S(H)$ чередующаяся цепь (2.3) естественным образом соответствует некоторой увеличивающей цепи относительно потока f_M в $S(H)$, а именно цепи

$$s,\ \langle s,\ x_0\rangle,\ x_0,\ \langle x_0,\ y_1\rangle,\ y_1,\ \langle x_1,\ y_1\rangle,\ x_1,\ \langle x_1,\ y_2\rangle,\ y_2,\ \ldots$$
$$\ldots,\ y_k,\ \langle x_k,\ y_k\rangle,\ x_k,\ \langle x_k,\ y_{k+1}\rangle,\ y_{k+1},\ \langle y_{k+1},\ t\rangle,\ t,$$

причём увеличение потока f_M вдоль этой цепи дает поток, соответствующий паросочетанию, определённому симметрической разностью $M \oplus P$. Это проиллюстрировано на рис. 2.2.

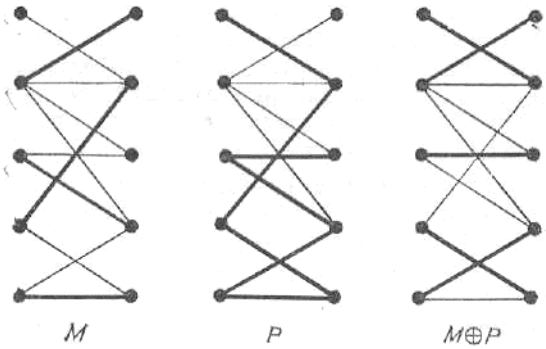


Рисунок 3.2 – Увеличение паросочетания M вдоль чередующейся цепи P

Из приведенных выше рассуждений непосредственно вытекает следующая теорема.

Теорема 2.2 (Берж). Паросочетание M в двудольном графе H наибольшее тогда и только тогда, когда в H не существует чередующейся цепи относительно M.

Рассмотрим, как выглядит псевдомаксимальный поток во вспомогательной бесконтурной сети, построенной для сети $S(H)$ и некоторого целочисленного, а, следовательно, нуль-единичного потока в этой сети. Потенциал каждой вершины, отличной от s и t, составляет 0 или 1, так как входной потенциал каждой вершины $x \in X$ и выходной потенциал каждой вершины $y \in Y$ во вспомогательной сети равен единице. Нетрудно заметить, что «увеличивающая кратная цепь», найденная на каждой итерации главного цикла процедуры *MAXPSA*, имеет вид одиночной увеличивающей цепи. Более того, все промежуточные вершины (т.е. отличные от s и t) такой цепи имеют потенциал, равный единице, а, следовательно, при увеличении потока вдоль этой цепи их потенциал убывает до нуля и они удаляются из сети. Теперь ясно, что во вспомогательной бесконтурной сети длины $l + 2$ поток

идет вдоль максимального множества путей длины $l + 2$ из s в t с попарно непересекающимися множествами промежуточных вершин. Этому множеству естественным образом соответствует максимальное множество чередующихся цепей длины l с попарно непересекающимися множествами вершин (под «максимальным» мы подразумеваем такое множество, которое нельзя увеличить на дополнительную чередующуюся цепь длины l и на множество вершин, не содержащее ни одной вершины прежних цепей).

Ниже детально описан **алгоритм 3.1**, носящий имя Хопкрофта-Карпа. В этом алгоритме вспомогательная бесконтурная сеть, а точнее вспомогательный бесконтурный граф, поскольку пропускные способности всех рёбер равны единице, строится при помощи процедуры *PGA*. Чтобы объяснить её действие, удобно ввести в рассмотрение для данного двудольного графа $H = \langle X,\ Y,\ E \rangle$ и паросочетания M в H ориентированный граф H_M посредством ориентации всех ребер $e \in M$ от Y к X и всех ребер $e \in E \setminus M$ от X к Y (пути в H_M от свободной вершины в X до свободной вершины в Y в точности соответствуют чередующимся цепям относительно M) . Процедура *PGA* сначала строит дуги от источника s до всех свободных вершин $x \in X$ (строки 6-10). Далее проводится поиск в ширину в графе H_M , начиная от свободных вершин в X . При нахождении каждой свободной вершины $y \in Y$ добавляем во вспомогательный бесконтурный граф дугу $\langle y,\ t \rangle$ (строки 14-17), причем с момента добавления первой такой дуги мы уже в процессе поиска не рассматриваем вершин, находящихся на расстоянии, большем или равном расстоянию от s до t (см. условие в строке 20). Увеличение существующего паросочетания вдоль максимального множества кратчайших увеличивающих цепей с попарно различными множествами вершин реализуется процедурой *ФАЗА*. Она ведёт поиск в глубину во вспомогательном бесконтурном графе, начиная с s. Каждый раз, когда в нашем процессе мы достигаем стока t (см. строку 50), содержимым стека *СТЕК* является последовательность вершин, представляющая некоторую чередующуюся цепь (отметим, что сток t не помещается в *СТЕК*, тем самым все время *НОВЫЙ*[t] = **истина** и, следовательно, t может посещаться многократно). Цикл 52 выполняет модификацию массива *СОЧЕТ*, соответствующего увеличению паросочетания вдоль такой цепи. Отметим, что если во время выполнения нашей процедуры некоторой вершине $v \in V$ приписывается значение *НОВЫЙ* [v]:= **ложь**, то эта процедура либо находит затем чередующуюся цепь, проходящую через эту вершину, либо устанавливает, что не существует ни одной цепи, проходящей через эту вершину и не пересекающей ни одной из ранее найденных цепей. Отсюда легко следует, что процедура *ФАЗА* находит максимальное множество непересекающихся кратчайших чередующихся цепей.

Корректность всего алгоритма следует из анализа метода Диница.

Алгоритм 2.1. *Нахождение наибольшего паросочетания методом Хопкрофта-Карпа*.

Данные: Двудольный граф $H = \langle X, Y, E \rangle$, представленный списками инцидентности *ЗАПИСЬ*[*x*], $x \in X$.

Результаты: Наибольшее паросочетание, представленное массивом *СОЧЕТ* (*СОЧЕТ*[*v*] есть вершина, сочетающаяся с *v* или с нулем, если *v* – свободная вершина).

```
procedure PGA;
  (* построение вспомогательного бесконтурного графа; переменные V, XV, X, Y, СОЧЕТ, ЗАПИСЬ,
  ХЗАПИСЬ, s, t – глобальные *)
begin
  for u ∈ V ∪ {s, t) do (* инициализация *)
    begin ДЛИНА [u] := ∞; ХЗАПИСЬ [u] :=∅
    end;
  (* поместить свободные вершины x ∈ X в очередь и в список ХЗАПИСЬ [s]*)
  ОЧЕРЕДЬ := ∅; XV := {s}; ДЛИНА [s] := 0;
  for x ∈ X do
    if СОЧЕТ [x] = 0 then (* x– свободная *)
      begin ОЧЕРЕДЬ⇐x;
      ХЗАПИСЬ [s] := ХЗАПИСЬ [s] ∪ {x}; ДЛИНА [x] := 1
      end;
  (* поиск в ширину, начиная со свободных вершин в X *)
  while ОЧЕРЕДЬ ≠∅ do
    begin u⇐ОЧЕРЕДЬ; XV:= XV∪{u};
      if u ∈ Y then
        if СОЧЕТ [u] = 0 then (* u – свободная, добавить дугу ⟨u, t⟩ *)
          begin ХЗАПИСЬ [u] := ХЗАПИСЬ [u]∪ {t}; XV := XV ∪ {t};
            ДЛИНА [t] : = ДЛИНА [u] + 1
          end
        else (*СОЧЕТ [u] ≠ 0*)
          begin x:= СОЧЕТ [u];
            if ДЛИНА [t] =∞ then (* добавить дугу ⟨u, x⟩ *)
              begin ОЧЕРЕДЬ⇐x; ХЗАПИСЬ [u] := ХЗАПИСЬ [u]∪{x};
                ДЛИНА[x] = ДЛИНА[u] + 1
              end
          end
      else (*u∈X*)
        for y∈ ЗАПИСЬ [u] do
          if ДЛИНА[u]< ДЛИНА\y\ then {* ДЛИНА[y]*= ДЛИНА[u]+1 или ∞ *)
            begin if ДЛИНА[y] = ∞ then ОЧЕРЕДЬ⇐y; (* y – новая *)
              ХЗАПИСЬ [u] := ХЗАПИСЬ [u] ∪ {y}; ДЛИНА [y] ;= ДЛИНА [u] + 1
            end
    end
end; (*PGA*)
procedure ФАЗА;
  (* увеличение существующего паросочетания вдоль максимального множества путей из s в t с
    попарно различными множествами промежуточных вершин во вспомогательном бесконтурном
    графе; переменные XV, НАЧАЛО, ХЗАПИСЬ, СОЧЕТ, s, t – глобальные*)
begin
  for u∈XV do (* инициализация *)
    begin НОВЫЙ [u] := истина;
      P[u]:=НАЧАЛО[u] {* НАЧАЛО [u] = указатель на начало списка ХЗАПИСЬ [u];
      P [u] – указатель на фактически анализируемый элемент списка ХЗАПИСЬ [u] *)
    end;
```

```
      (*поиск в глубину во вспомогательном бесконтурном графе, начиная с источника s *)
39    СТЕК:=∅; СТЕК⇐s; НОВЫЙ [s] := ложь;
40    while СТЕК Ф 0 do (* главный цикл *)
41      begin u:=top (СТЕК); (*u = верхний элемент стека*)
42        (* отыскание первой новой вершины в списке ХЗАПИСЬ[u]*)
43        if P[u] = niI then b:= ложь else b: = not НОВЫЙ [P[u]↑. верш];
44        while b do
45          begin P[u]:=P[u]↑. след;
46            if P [u] = nil then b := ложь
47            else b:= not НОВЫЙ [P[u]↑. верш]
48          end;
49        if P [u] ≠ nil then (* найдена новая вершина*)
50          if |P[u]↑. верш = t then (* найдена чередующаяся цепь *)
51          (*увеличение существующего паросочетания вдоль чередующейся цепи, хранящейся в стеке*)
52           while top (СТЕК) ≠ s do
53            begin y⇐СТЕК; x⇐СТЕК;
54              СОЧЕТ[x]:= y; СОЧЕТ[y]:=x
55            end
56          (* в стеке остался только источник s*)
57        else (*|P[u]↑. верш ≠ t*)
58          begin v:=P[u]↑. верш; СТЕК⇐v; НОВЫЙ [v]:= ложь
59          end
60      else (P[u] = nil, т. е. в списке ХЗАПИСЬ[u] уже нет новых вершин *)
61          u ⇐ СТЕК (*удалить верхний элемент стека*)
62      end
63 end; (*ФАЗА*)
64 begin (* главная программа*)
65   for v∈V do СОЧЕТ [v] :== 0; (* инициализация; паросочетание – пустое *)
66   PGA;   (* построение   вспомогательного бесконтурного графа *)
67   repeat ФАЗА; PGA
68   until not XV [t]
69 end
```

Большая эффективность алгоритма Хопкрофта-Карпа объясняется тем, что порядок числа фаз есть $\sqrt{n}$. Для доказательства ограниченности числа фаз нам будет необходима следующая лемма.

Лемма 2.1. Пусть M è N – два паросочетания в двудольном графе $H = \langle X,\ Y,\ E \rangle$, и пусть $|M| = r < s = |N|$. Тогда симметрическая разность $M \oplus N$ содержит не менее $s - r$ чередующихся цепей относительно M с попарно различными множествами вершин.

Доказательство. Рассмотрим граф $H^* = \langle X,\ Y,\ M \oplus N \rangle$ и обозначим через $C_1,\ \ldots,\ C_p$ компоненты связности этого графа. Каждая вершина графа H^* принадлежит не более чем одному ребру из $M \setminus N$ и не более чем одному ребру из $N \setminus M$ (так как M è N – паросочетания). Отсюда следует, что каждая компонента связности C_i имеет один из трёх следующих видов:

(1) изолированная вершина;

(2) цикл чётной длины с рёбрами попеременно из $M \setminus N$ и $N \setminus M$;

(3) цепь с рёбрами попеременно из $M \setminus N$ и $N \setminus M$ (это не обязательно чередующаяся цепь: оба ее конца могут принадлежать X или оба Y).

Обозначим через E_i множество рёбер компоненты C_i и рассмотрим величину $\delta_i = |E_i \cap N| - |E_i \cap M|$. Имеем $\delta_i \in \{-1,\ 0,\ 1\}$, причём $\delta_i = 1$ тогда и только тогда, когда C_i – чередующаяся цепь относительно

М. Более того, $\delta_i = 1$ для не менее чем $s - r$ указателей, так как

$$\sum_{i=1}^{p} \delta_i = \sum_{i=1}^{p} \left(\left| E_i \cap N \right| - \left| E_i \cap M \right| \right) = \sum_{i=1}^{p} \left| E_i \cap N \right| - \sum_{i=1}^{p} \left| E_i \cap M \right| =$$
$$= \left| N \setminus M \right| - \left| M \setminus N \right| = \left| N \right| - \left| M \right| = s - r.$$

Этим доказательство леммы закончено. ■

Теперь мы можем приступить к доказательству существования объявленного ограничения на число фаз алгоритма Хопкрофта-Карпа.

Теорема 2.2. Число фаз алгоритма Хопкрофта-Карпа не превышает $2\left\lfloor \sqrt{s} \right\rfloor + 1$, где s – наибольшая мощность паросочетания в данном графе.

Доказательство. Обозначим через P_0, P_1, …, P_{s-1} последовательные чередующиеся цепи, построенные алгоритмом, и определим $M_0 = \varnothing$, $M_i = M_{i-1} \oplus P_{i-1}$ для $1 \le i \le s$. Согласно алгоритму P_i является чередующейся цепью относительно M_i, и мы уже знаем, что $\left|P_0\right| \le \left|P_1\right| \le \ldots, \le \left|P_{s-1}\right|$ (см. лемму 4.4). Число фаз – это не что иное, как количество различных чисел в последовательности $\left|P_0\right|, \ldots, \left|P_{s-1}\right|$. Обозначим $r = \left\lfloor s - \sqrt{s} \right\rfloor$, и пусть r^* – такой наименьший индекс, что $\left|P_{r^*}\right| = \left|P_r\right|$. Согласно построению вспомогательного бесконтурного графа P_{r^*} – кратчайшая чередующаяся цепь относительно M_{r^*} (фактически P_i – это кратчайшая чередующаяся цепь относительно M_i для каждого *i*). По **лемме 2.1** существует по крайней мере $\left|M_s\right| - \left|M_{r^*}\right| = s - r^*$ непересекающихся чередующихся цепей относительно M_{r^*}, следовательно, кратчайшая чередующаяся цепь относительно M_{r^*} содержит не более чем $r^* \big/ \left(s - r^*\right)$ рёбер паросочетания M_{r^*}.

Отсюда

$$\left|P_r\right| = \left|P_{r^*}\right| \le \frac{2r^*}{s - r^*} + 1 \le \frac{2r}{s - r} + 1 = \frac{2\left\lfloor s - \sqrt{s} \right\rfloor}{\left\lceil \sqrt{s} \right\rceil} + 1 =$$
$$= \frac{2s}{\left\lceil \sqrt{s} \right\rceil} - 1 \le 2\sqrt{s} - 1 \le 2\left\lfloor \sqrt{s} \right\rfloor + 1.$$

Длина каждой цепи нечётная, а, следовательно, последовательность $\left|P_0\right|, \ldots, \left|P_r\right|$ содержит не более чем $\left\lfloor \sqrt{s} \right\rfloor + 1$ различных чисел. Последовательность $\left|P_{r+1}\right| \le \ldots, \le \left|P_{s-1}\right|$ может содержать не более чем $\left(s-1\right) - \left(r+1\right) + 1 = s - r - 1 = \left\lceil \sqrt{s} \right\rceil - 1 \le \left\lfloor \sqrt{s} \right\rfloor$ других чисел, что в сумме даёт требуемое ограничение $2\left\lfloor \sqrt{s} \right\rfloor + 1$. ■

Теперь легко оценить сложность всего алгоритма 2.1. Как поиск в ширину в процедуре *PGA*, так и поиск в глубину в процедуре *ФАЗА* требуют $O\left(m + n\right)$ шагов, что при числе фаз порядка $\sqrt{n}$ даёт общую сложность $O\left(\left(m + n\right)\sqrt{n}\right)$ или, если использовать только число вершин в качестве размерности задачи, $O\left(n^{5/2}\right)$.

Стоит отметить, что известен алгоритм построения наибольшего паросочетания в произвольном необязательно двудольном графе со сложностью $O\left(n^{5/2}\right)$. Однако этот алгоритм

несравненно более сложный.

Вопросы для самопроверки

1. Какие графы называются двудольными ?
2. Что собой представляет паросочетание в неориентированном графе $G = \langle V, E \rangle$?
3. Что понимается под термином чередующаяся цепь ?
4. Когда паросочетание в двудольном графе наибольшее ?

Задание на лабораторную работу

1. По заданному алгоритму 2.1 написать и отладить программу нахождения наибольшего паросочетания методом Хопкрофта-Карпа.
2. Продемонстрировать работающую программу преподавателю и получить отметку о ее выполнении.
3. Провести анализ полученного алгоритма на сложность.
4. Оформить отчёт о проделанной работе. Он должен включать в себя:

– цель работы;

– ответы на вопросы для самопроверки;

– схему обрабатывающего алгоритма и описание его работы;

– распечатки экранных форм, полученных в результате работы программы.

Лабораторная работа № 3
Жадные алгоритмы для матричного матроида и матроида трансверсалей

Цель работы – написать и отладить программу генерирования всех k-элементных подмножеств множества $\{1, ..., n\}$ в лексикографическом порядке.

Теоретическая часть

Жадные алгоритмы решения оптимизационных задач. Рассмотрим следующую матрицу с действительными неотрицательными коэффициентами:

$$\mathbf{A} = \begin{pmatrix} 7 & 5 & 1 \\ 3 & 4 & 3 \\ 2 & 3 & 1 \end{pmatrix}.$$

Решим следующую оптимизационную задачу.

Задача 1. Найти такое подмножество элементов матрицы, что

(*а*) в каждом столбце находится не более одного выбранного элемента и

(*б*) сумма выбранных элементов является наибольшей из возможных.

Попробуем решить эту задачу следующим образом: будем выбирать элементы последовательно, причём каждый раз будем выбирать наибольший из элементов, которые мы можем добавить, не нарушая условия (*а*). Будем действовать так вплоть до момента, пока добавление произвольного элемента не нарушит условия (*а*). Алгоритм такого типа мы будем называть *жадным*.

Для задачи 1 и матрицы $\mathbf{A}$ жадный алгоритм находит подмножество

$$\begin{pmatrix} 7 & \hat{5} & 1 \\ 3 & 4 & \hat{3} \\ 2 & 3 & 1 \end{pmatrix}$$

которое действительно даёт наибольшую возможную сумму.

Нетрудно отметить, что с помощью жадного алгоритма правильно находится решение задачи 1 для произвольной вещественной матрицы с неотрицательными элементами.

Рассмотрим несколько иную задачу.

Задача 2. Найти такое подмножество элементов матрицы, что

(*а*) в каждом столбце и в каждой строке находится не более одного выбранного элемента и

(*б*) сумма выбранных элементов является наибольшей из возможных.

На этот раз применение жадного алгоритма к задаче 2 и матрице $\mathbf{A}$ даёт подмножество

$$\begin{pmatrix} 7 & 5 & 1 \\ 3 & 4 & 3 \\ 2 & 3 & \hat{1} \end{pmatrix}$$

с суммой 12, что не является правильным решением, поскольку подмножество

$$\begin{pmatrix} 7 & 5 & 1 \\ 3 & 4 & \hat{3} \\ 2 & \hat{3} & 1 \end{pmatrix}$$

имеет сумму 13. Следовательно, на втором шаге не следовало бы быть жадным, мы выигрываем в конечном результате, выбирая несколько меньший элемент (3 вместо 4).

Возникает вопрос: когда выгодно быть жадным? Сформулируем это более строго.

Мы будем рассматривать оптимизационные задачи следующего типа.

Задача 3. Даны: конечное множество E, семейство его подмножеств $\mathrm{I} \subseteq \mathrm{P}(E)$ и функция $w : E \to R^+$, где R^+ обозначает множество вещественных неотрицательных чисел. Найти подмножество $S \in \mathrm{I}$ с наибольшей суммой $\sum_{e \in S} w(e)$.

Как задача 1, так и задача 2 являются частными случаями задачи 3. В обоих случаях E есть множество позиций матрицы, а w ставит в соответствие позиции $\langle i, j \rangle$ матрицы $\left[a_{i,j}\right]$ число $a_{i,j}$. Для задачи 1

$S \in \mathrm{I} \Leftrightarrow$ каждый столбец содержит не более одной позиции из множества S,

а в случае задачи 2

$S \in \mathrm{I} \Leftrightarrow$ каждый столбец и каждая строка содержит не более одной позиции из множества S.

Теперь мы можем сформулировать наш вопрос следующим образом: при каких условиях относительно семейства I жадный алгоритм правильно решает задачу 3 для произвольной функции w?

Оказывается, что на этот вопрос можно найти простой ответ. А именно, достаточно, чтобы пара $\langle E, \mathrm{I} \rangle$ образовывала так называемый матроид.

Матроиды и их основные свойства. Матроиды были введены Уитни в работе [73] в совершенно ином контексте, нежели жадные алгоритмы, а именно в исследованиях абстрактной теории линейной зависимости. Существует много эквивалентных определений матроида. Для нас наиболее удобным будет следующее определение:

Матроидом мы будем называть произвольную пару $M = \langle E, \mathrm{I} \rangle$, где E – конечное множество, а $\mathrm{I} \subseteq \mathrm{P}(E)$ – семейство, удовлетворяющее условиям:

$M1$ $\varnothing \in \mathrm{I}$ и если $A \in \mathrm{I}$ è $B \subseteq A$, òî $B \in \mathrm{I}$.

$M2$ Для произвольных $A, B \in \mathrm{I}$, таких что $|B| = |A| + 1$,

существует элемент $e \in B \setminus A$, такой что $A \cup \{e\} \in \mathrm{I}$.

(Условие $\varnothing \in \mathrm{I}$ исключает вырожденный случай $\mathrm{I} = \varnothing$.)

Множества семейства I мы будем называть *независимыми множествами*, а остальные подмножества из $\mathrm{P}(E) \setminus \mathrm{I}$ – *зависимыми множествами* матроида M. В этом проявляется уже упомянутая связь матроидов с теорией линейной зависимости: аксиомы матроидов выбраны таким образом, чтобы они отражали наиболее характерные свойства независимых множеств линейного пространства, точнее говоря, независимых подмножеств, содержащихся в некотором конечном подмножестве этого пространства (вспомним, что подмножество $\{e_1, \ldots, e_n\}$ линейного пространства называется независимым, если не существует набора скаляров $\lambda_1, \ldots, \lambda_n$, не всех равных нулю, такого что $\lambda_1 e_1 + \ldots + \lambda_n e_n = 0$). В самом деле, очевидно, что произвольное подмножество независимого множества линейного пространства независимо. Если $|B| = |A| + 1$ для линейно независимых подмножеств A, B линейного пространства, то A порождает пространство размерности $|A|$, которое может содержать не более чем $|A|$ элементов множества B. Следовательно, существует элемент $e \in B \setminus A$, не принадлежащий этому подпространству. Множество $A \cup \{e\}$ порождает подпространство размерности $|A| + 1$ и, следовательно, является линейно независимым.

Будем говорить, что два матроида $\langle E, \mathrm{I} \rangle$ и $\langle E', \mathrm{I}' \rangle$ *изоморфны*, если существует взаимно однозначное отображение f множества E на множество E', такое что $A \in \mathrm{I}$ тогда и только тогда, когда $f(A) \in \mathrm{I}'$. Часто мы не будем делать различия между изоморфными матроидами.

Для произвольного подмножества $C \subseteq E$ мы будем изучать его максимальные независимые подмножества, т.е. независимые подмножества $A \subseteq C$, обладающие таким свойством: не существует независимого подмножества B, такого что $A \subset B \subseteq C$.

Теорема 3.1. Пусть E – конечное множество, I – семейство его подмножеств, удовле-

творяющих условию $M1$. При этих предположениях $M = \langle E, \mathrm{I} \rangle$ является матроидом тогда и только тогда, когда удовлетворяется условие

$M3$ Для произвольного подмножества $C \subseteq E$ каждые два максимальных подмножества множества C имеют одинаковую мощность.

Доказательство. Предположим, что $M = \langle E, \mathrm{I} \rangle$ является матроидом и для некоторого $C \subseteq E$ существуют два максимальных независимых подмножества $A, B \subseteq C$, такие что $|B| \geq |A|$. Выберем произвольное подмножество $B' \subseteq B$ (независимое по условию $M1$!), такое что $|B'| = |A| + 1$. В силу $M2$ существует такой элемент $e \in B' \setminus A \subseteq C$, что $A \cup \{e\} \in \mathrm{I}$ вопреки максимальности множества A.

Напротив, предположим, что выполняются условия $M1$ и $M3$, Выберем такие произвольные подмножества $A, B \in \mathrm{I}$, что $|B| = |A| + 1$ и обозначим $C = A \cup B$. Допустим, что не существует такого элемента $e \in B \setminus A$, что $A \cup \{e\} \in \mathrm{I}$. Это означало бы, что A является максимальным независимым подмножеством множества C. Расширяя B до максимального независимого подмножества $B^* \subseteq C$, мы имели бы $|A| < |B^*|$ вопреки условию $M3$. Следовательно, условия $M1$ и $M3$ влекут за собой условие $M2$. ■

Из **теоремы 3.1** следует, что условия $M1$ и $M3$ образуют эквивалентную систему аксиом для матроидов.

Мощность максимального подмножества множества $C \subseteq E$ называется *рангом* этого множества и обозначается через $r(C)$:

$$r(C) = \max\{|A| : A \in \mathrm{I} \ \wedge A \subseteq C\}.$$

Очевидно, что подмножество $C \subseteq E$ является независимым тогда и только тогда, когда $r(C) = |C|$. Каждое максимальное независимое множество матроида $M = \langle E, \mathrm{I} \rangle$ будем называть (по аналогии с линейными пространствами) *базой* этого матроида, а ранг $r(E)$, являющийся аналогом размерности линейного пространства, *рангом матроида*. Отметим важное следствие теоремы 4.1.

Следствие 3.1. Каждые две базы матроида имеют одинаковое число элементов. ■

Отметим также, что каждое независимое множество $C \in \mathrm{I}$ можно расширить до базы $B \supseteq C$; достаточно поочередно добавлять в C новые элементы, присоединение которых не нарушает независимости, вплоть до момента, когда таких элементов больше не существует. Полученное множество будет максимальным независимым множеством, а, следовательно, базой. Аналогично, произвольное независимое множество $A \subseteq C$ можно расширить до максимального независимого подмножества множества C.

Отметим некоторые свойства рангов.

Теорема 3.2. Для произвольных $A, B \subseteq E$ è $e, f \in E$ имеем

$R1$ $0 \le r(A) \le |A|$,

$R2$ если $A \subseteq B$, то $r(A) \le r(B)$,

$R3$ $r(A \cup B) + r(A \cap B) \le r(A) + r(B)$,

$R4$ $r(A) \le r(A \cup \{e\}) \le r(A) + 1$,

$R5$ если $r(A \cup \{e\}) = r(A \cup \{f\}) = r(A)$, то $r(A \cup \{e, f\}) = r(A)$.

Доказательство. Свойства R1 и R2 очевидны. Докажем R3. Пусть $\{e_1, \ldots, e_p\}$ – максимальное независимое множество в $A \cap B$. Расширим его до максимального независимого множества $\{e_1, \ldots, e_p, f_1, \ldots, f_q\} \subseteq A$, а затем до максимального независимого множества $\{e_1, \ldots, e_p, f_1, \ldots, f_q, g_1, \ldots, g_r\} \subseteq A \cup B$. Имеем $p = r(A \cap B)$, $p + q = r(A)$, $p + r \le$ $\le p + q + r = r(A \cup B)$, а, следовательно,

$$r(A \cup B) + r(A \cap B) = (p + q + r) + p = (p + q) + (p + r) \le r(A) + r(B).$$

Условие $R4$ очевидно. $R5$ вытекает из $R3$:

$$r(A) \le r(A \cup \{e, f\}) = r(A \cup \{e\}) + r(A \cup \{f\}) - r((A \cup \{e\}) \cap (A \cup \{f\})) =$$
$$= r(A) + r(A) - r(A) = r(A). \qquad \blacksquare$$

Продолжая аналогию с линейными пространствами, будем говорить, что элемент e *зависим* от множества A, если $r(A \cup \{e\}) = r(A)$, и будем обозначать через $sp(A)$ множество всех элементов, зависимых от A:

$$sp(A) = \{e \in E : r(A \cup \{e\}) = r(A)\}.$$

Множество $A \subseteq E$ называется *подпространством* матроида, если $A = sp(A)$, т. е. если $r(A \cup \{e\}) = r(A) + 1$ для произвольного $e \in E \setminus A$.

Теорема 3.3. Для произвольных $A, B \subseteq E$ è $e, f \in E$ имеем

$S1$ $A \subseteq sp(A)$,

$S2$ åñëè $A \subseteq B$, òî $sp(A) \subseteq sp(B)$,

$S3$ $sp(sp(A)) = sp(A)$,

$S4$ åñëè $f \notin sp(A)$ è $f \in sp(A \cup \{e\})$, òî $e \in sp(A \cup \{f\})$. ■

Доказательство. Условие $S1$ очевидно: если $e \in A$, то $r(A \cup \{e\}) = r(A)$, т. е. $e \in sp(A)$. Для доказательства $S2$ воспользуемся $R3$. Предположим, что $A \subseteq B$ и $e \in sp(A)$. Тогда

$$r(B) \le r(B \cup \{e\}) = r((A \cup \{e\}) \cup B) \le r(A \cup \{e\}) + r(B) - r(A \cup (B \cap \{e\})) = r(A) + r(B) - r(A) = r(B)$$

и, следовательно, $e \in sp(B)$.

Из условий $S1$ и $S2$ следует, что $sp(A) \subseteq sp(sp(A))$. Докажем теперь противоположное включение. Для этого нам будет необходимо равенство

$$r(sp(A)) = r(A). \tag{4.1}$$

Чтобы доказать его, предположим, что $sp(A) \setminus A = \{e_1, \ldots, e_k\}$. Из определения $sp(A)$ имеем $r(A \cup \{e_i\}) = r(A)$, $i = 1, \ldots, k$. Предположим, что для некоторого $i < k$ имеет место

$r\left(A\cup\{e_1,\ \ldots,\ e_i\}\right)=r(A)$. Тогда из $R3$ получаем

$$r\left(A\cup\{e_1,\ \ldots,\ e_i,\ e_{i+1}\}\right)=r\left(\left(A\cup\{e_1,\ \ldots,\ e_i\}\right)\cup\left(A\cup e_{i+1}\right)\right)\le$$

$$\le r\left(A\cup\{e_1,\ \ldots,\ e_i\}\right)+r\left(A\cup e_{i+1}\right)-r(A)=r(A)+r(A)-r(A)=r(A).$$

Отсюда индукцией по i получаем требуемое равенство (3.1). Доказательство включения $sp\left(sp(A)\right)\subseteq sp(A)$ теперь просто. Предполагая $e\in sp\left(sp(A)\right)$, т.е. $r\left(sp(A)\cup\{e\}\right)=r\left(sp(A)\right)$, имеем $r(A)\le r\left(A\cup\{e\}\right)\le r\left(sp(A)\cup\{e\}\right)=r\left(sp(A)\right)=r(A)$ т.е. $e\in sp(A)$.

Наконец, свойство $S4$ вытекает из определения зависимости элемента от множества: предположив $f\notin sp(A)$ и $f\in sp\left(A\cup\{e\}\right)$, имеем

$$r(A)+1=r\left(A\cup\{f\}\right)\le r\left(A\cup\{e,\ \ f\}\right)=r\left(A\cup\{e\}\right)\le r(A)+1,$$

а отсюда $r\left(A\cup\{f\}\right)=r\left(A\cup\{e,\ \ f\}\right)$, т. е. $e\in sp\left(A\cup\{f\}\right)$. ■

Отметим, что из свойства $S3$ следует, что $sp(A)$ является подпространством матроида. Будем называть его *подпространством, натянутым на множество* A.

Последним важным понятием данного раздела является понятие цикла. *Циклом* матроида мы будем называть каждое минимальное зависимое множество, т.е. такое зависимое множество C, что $C\setminus\{e\}$ является независимым для произвольного $e\in C$. Понятие цикла имеет особенно ясную интерпретацию для случая графовых матроидов, о которых речь пойдёт в разд. 5.5.

Теорема 3.4. Для произвольных циклов C, D выполняются условия:

$C1$ Если $C\subseteq D$, то $C=D$,

$C2$ Если $C\ne D$ и $e\in C\cap D$, то существует цикл $F\subseteq(C\cup D)\setminus\{e\}$.

Доказательство. Условие $C1$ выражает просто свойство минимальности, содержащееся в определении цикла. Докажем теперь $C2$. Множества $C\setminus\{e\}$ и $D\setminus\{e\}$ независимые, а следовательно, $r\left(C\setminus\{e\}\right)=|C|-1$ и $r\left(D\setminus\{e\}\right)=|D|-1$. В силу условия $R3$

$$r(C\cup D)+r(C\cap D)\le r(C)+r(D)=|C|+|D|-2=|C\cup D|+|C\cap D|-2. \tag{4.2}$$

Множество $C\cap D$ независимое, так как оно является собственным подмножеством обоих циклов. Отсюда $r(C\cap D)=|C\cap D|$ и неравенство (4.2) можно переписать как

$$r(C\cup D)\le|C\cup D|-2. \tag{4.3}$$

Предположим теперь, что не существует цикла F, о котором идёт речь в условии $C2$. Тогда множество $(C\cup D)\setminus\{e\}$ независимое, $r\left((C\cup D)\setminus\{e\}\right)=|C\cup D|^{-1}$ и в результате $r(C\cup D)\ge|C\cup D|-1$ вопреки неравенству (4.3).

Отметим важное следствие из свойства $C2$.

Следствие 3.2. Если A – независимое множество, то для произвольного $e\in E$ множество $A\cup\{e\}$ содержит не более одного цикла.

Доказательство. Если бы существовали два различных цикла $C,D\subseteq A\cup\{e\}$, то очевидно, что мы имели бы $e\in C\cap D$, и вследствие свойства $C2$ существовал бы цикл $F\subseteq(C\cup D)\setminus\{e\}\subseteq A$ вопреки пред-

положению о независимости множества A.

Свойства матроидов, рассмотренные выше, связаны с линейно независимыми множествами векторов линейного пространства. Приведём два других примера матроидов.

Для произвольного конечного множества E пара $\langle E, \mathrm{P}(E)\rangle$, очевидно, удовлетворяет условиям $M1$ и $M2$. Будем называть ее *свободным матроидом* на множестве E. В таком матроиде каждое множество $A \subseteq E$ независимое, и в результате $r(A) = |A|$.

Другой пример получаем, рассматривая произвольное разбиение $\pi = \{D_1, \ldots, D_k\}$ множества E и определяя

$$\mathrm{I} = \{A \subseteq E : |A \cap D_i| \le 1 \text{ для } i = 1, \ldots, k\}.$$

Отметим, что если $A, B \in \mathrm{I}$ и $|B| = |A| + 1$, то должен существовать такой индекс i, что $D_i \cap A \in \varnothing$, í î $D_i \cap B = \varnothing$. Элемент $e \in D_i \cap B$ можно «перенести» в A и получить множество $A \cup \{e\} \in \mathrm{I}$. Это доказывает справедливость условия $M2$. Матроид $M = \langle E, \mathrm{I}\rangle$, определённый таким образом, будем называть *матроидом разбиений*.

Теорема Радо-Эдмондса. Вернёмся к жадным алгоритмам. Сформулируем сначала общую схему алгоритмов этого типа. Мы будем рассматривать конечное множество E, функцию $w : E \to R^+$ и семейство $\mathrm{I} \subseteq \mathrm{P}(E)$. Значение $w(e)$ называется *весом* элемента e.

Вес подмножества $A \subseteq E$ определяется следующим образом:

$$w(A) = \sum_{e \in A} w(e).$$

Алгоритм 3.1. (*Жадный алгоритм*).

```
begin
упорядочить множество E по убыванию весов так, чтобы E = {e_1, ..., e_n}, где w(e_1) ≥ w(e_2) ≥ ... ≥ w(e_n);
   S := ∅;
      for i: = 1 to n do
      if S ∪ {e_i} ∈ I  then S := S ∪ {e_i}
end
```

Теорема 3.5 (Радо, Эдмондс). Если $M = \langle E, \mathrm{I}\rangle$ есть матроид, то множество S, найденное жадным алгоритмом, является независимым множеством с наибольшим весом. Напротив, если $M = \langle E, \mathrm{I}\rangle$ не является матроидом, то существует такая функция $w : E \to R^+$, что S не будет независимым множеством с наибольшим весом.

Следует обратить внимание на один факт. Поскольку мы определили вес подмножества

как сумму весов его элементов, то могло бы казаться, что оптимальное множество S будет существенным образом зависеть от численных значений весов отдельных элементов. Однако же эта зависимость очень слаба: множество S зависит только от упорядочения весов отдельных элементов. В жадном алгоритме после сортировки элементов веса совершенно перестают нас интересовать.

Стоит отметить, что множество S оптимальное в очень сильном смысле. У него не только максимальна сумма элементов, но также в любом независимом множестве T вес i-го по величине элемента не больше веса i-го по величине элемента в S. Этот факт мы будем называть *оптимальностью по Гейлу*. Таким образом, в матроиде нельзя выбрать независимое множество, состоящее из «меньшего числа больших по величине (по весу) элементов».

Отметим также, что при обращении упорядочения элементов жадный алгоритм выберет множество S, которое не только имеет наименьший вес, но и i-й по величине элемент (считая от наименьшего) будет не больше i-го по величине элемента произвольной базы. Теперь мы можем посмотреть на применение жадного алгоритма к задаче 1 с общей точки зрения: мы имели там дело с матроидом разбиений, порождённым разбиением позиций матрицы на столбцы.

Матричные матроиды

Матричные матроиды отличаются от матроидов, определяемых независимыми подмножествами линейного пространства, только видом. Пусть $\{v_1, \ldots, v_n\}$ – элементы некоторого линейного пространства размерности m, и пусть $\{b_1, \ldots, b_m\}$ – базис этого пространства. Векторы $v_1, \ldots, v_n$ имеют однозначно определённое разложение относительно базиса $b_1, \ldots, b_m$:

$$
\begin{aligned}
v_1 &= a_{11}b_1 + a_{21}b_2 + \ \ldots\ + a_{m1}b_m, \\
v_2 &= a_{12}b_1 + a_{22}b_2 + \ \ldots\ + a_{m2}b_m, \\
&\vdots \\
v_n &= a_{1n}b_1 + a_{2n}b_2 + \ \ldots\ + a_{mn}b_m.
\end{aligned}
$$

Рассмотрим матрицу

$$
\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}. \tag{3.4}
$$

Ее столбцы, обозначим их $e_1, \ldots, e_n$, соответствуют векторам $v_1, \ldots, v_n$, причём подмножество векторов линейно независимо тогда и только тогда, когда соответствующее им множество столбцов линейно независимо. Обратно, столбцы произвольной матрицы **A** вида (4.4) можно трактовать как векторы некоторого *m*-мерного линейного пространства. Каждая такая матрица определяет матроид $M(A) = \langle E, \mathrm{I} \rangle$, где E есть множество её столбцов, а $B \in \mathrm{I}$ тогда и только тогда, когда множество столбцов B линейно независимо. Матроид, определённый таким образом, будем называть *матроидом матрицы* **A**. Матроид называется *матричным*, если он является (изоморфен с) матроидом некоторой матрицы.

Алгоритм 3.2. (*Жадный алгоритм для матричного матроида*).

Данные: Матрица **A** с *m* строками и *n* столбцами, столбцы которой упорядочены по невозрастанию весов (веса неотрицательны).

Результаты: Независимое множество столбцов с наибольшей суммой весов (S содержит номера этих столбцов).

```
1  begin
2     S:= ∅;
3     for j:= 1 to n do
4       begin i :=0;
5          while (A [i, j] = 0) and (i < m) do i := i+ 1;
6          if A [i, j] ≠ 0 then (* j-й столбец ненулевой *)
7             begin S := S ∪ { j } ;
8               for k := j + 1to n do
9                 for l:= 1 to m do
10                   A [l, k] := A [l, k] – A [l, j] * A [i, k]/A [i, j)
11            end (*A [i, k] = 0 для j + 1 ≤ k ≤ n *])
12       end
13 end
```

Процесс преобразования матрицы **A**, реализуемый этим алгоритмом, – это не что иное, как известный из численного анализа метод исключения Гаусса. В каждой итерации цикла 3 алгоритм проверяет (строка 5), состоит ли *j*-й столбец из одних нулей. Если да ($A[i, j] = 0$ в строке 6), то очевидно, что *j*-й столбец не принадлежит ни к одному линейно независимому множеству столбцов. Если нет ($A[i, j] \neq 0$ в строке 6), то мы включаем *j*-й столбец в искомое множество линейно независимых столбцов и вычитаем для всех $k > j$ из *k*-го столбца *j*-й столбец, умноженный на $A[i, k] / A[i, j]$. Это не нарушает линейной зависимости столбцов, но приводит к обращению в нуль всех элементов *i*-й строки справа от $A[i, j]$ (легко увидеть, что последующие итерации цикла 3 не меняют положения дел). По окончании работы алгоритма множество S содержит номера ненулевых столбцов. Эти столбцы

линейно независимы, так как после соответствующей перестановки строк они содержат подматрицу размером $|S| \times |S|$ с нулями выше главной диагонали и ненулевыми элементами на диагонали.

Сложность алгоритма можно легко оценить, если заметить, что доминирующей частью $(O(nm)$ шагов) блока 4 является цикл 8. Этот блок выполняется n раз, что в сумме даёт $O(n^2 m)$ шагов.

Покажем теперь пример применения **алгоритма 3.2**, связанный с планированием экспериментов. Вообще говоря, будем рассматривать эксперименты, в которых некоторый объект одновременно подвергается действию многих независимых факторов, причём можно количественно измерить суммарное изменение объекта. Нашей задачей будет определение влияния отдельных факторов на изменение объекта. Примем линейную модель, в которой изменение объекта выражается формулой

$$b = c_1 x_1 + c_2 x_2 + \ \dots \ + c_m x_m,$$

где x_i есть интенсивность *i*-го фактора, а $c_1, \ \dots, \ c_m$ – коэффициенты, которые нужно определить. Проще всего было бы совершить m экспериментов, подвергая объект в *i*-м эксперименте действию только *i*-го фактора с единичной интенсивностью; тогда $x_i = 1, \ x_k = 0$ для $k \neq i$, что даёт возможность непосредственно определить c_i. Однако такое решение часто невозможно по чисто техническим причинам. Предположим, например, что мы наблюдаем влияние содержания различных минералов в почве на рост урожая некоторой культуры, причём мы имеем в своём распоряжении n $(n \geq m)$ удобрений, являющихся смесью этих минералов в строго определённых пропорциях. Уравнения, соответствующие возможным экспериментам, можно представить следующим образом;

$$\begin{aligned} & c_1 a_{11} + c_2 a_{21} + \ \dots \ + c_m a_{m1} = b_1, \\ & c_1 a_{12} + c_2 a_{22} + \ \dots \ + c_m a_{m2} = b_2, \\ & \vdots \\ & c_1 a_{1n} + c_2 a_{2n} + \ \dots \ + c_m a_{mn} = b_n. \end{aligned} \tag{3.5}$$

где a_{ij} обозначает количество *i*-го минерала в *j*-м эксперименте (мы предполагаем использование стандартных количеств удобрений и площадей культур). Рассмотрим матрицу $\mathbf{A} = \left[a_{ij} \right]$. Её столбцы соответствуют экспериментам (удобрениям), а строки – минералам. Каждое линейно независимое множество из m столбцов определяет множество экспериментов, которые нужно провести, чтобы вычислить коэффициенты $c_1, \ \dots, \ c_m$ из системы

уравнений (4.5). Если упорядочить столбцы по неубыванию стоимости соответствующих им экспериментов, то жадный алгоритм определит самую дешёвую систему экспериментов.

Матроиды трансверсалей

Пусть $\mathsf{A} = (A_1, \ldots, A_n)$ – семейство подмножеств (не обязательно различных) некоторого конечного множества E. Будем говорить, что множество $S \subseteq E$ является *частичной трансверсалью* семейства A, если существует такое инъективное отображение $\varphi : S \to \{1, \ldots, n\}$, что $e \in A_{\varphi(e)}$ для каждого $e \in S$. Это эквивалентно следующему утверждению: множество $S = \{e_1, \ldots, e_k\}$ является частичной трансверсалью семейства A, если для некоторой перестановки f множества $\{1, \ldots, k\}$ и для некоторой последовательности индексов $1 \le i_1 \le \ldots \le i_k \le n$ последовательность $\langle e_{f(1)}, \ldots, e_{f(k)} \rangle$ является системой различных представителей для последовательности $\langle A_{i_1}, \ldots, A_{i_k} \rangle$. Отметим, что для частичной трансверсали S семейства $(A_1, \ldots, A_n)$ существует, вообще говоря, много различных взаимно однозначных функций $\varphi : S \to \{1, \ldots, n\}$, удовлетворяющих условию $e \in A_{\varphi(e)}$, $e \in S$.

Основным результатом является следующая теорема Эдмондса и Фалкерсона.

Теорема 3.6. Пусть $\mathsf{A} = (A_1, \ldots, A_n)$ – семейство подмножеств конечного множества E, и пусть I – семейство частичных трансверсалей семейства A. Тогда $M(\mathsf{A}) = \langle E, \mathsf{I} \rangle$ является матроидом.

Доказательство. Очевидно, что условие $M1$ выполняется. Чтобы доказать выполнение условия $M2$, предположим, что $A, B \in \mathsf{I}$, $|A| = k$, $|B| = k+1$. Мы можем предполагать, что $A = \{a_1, \ldots, a_k\}$ и $B = \{b_1, \ldots, b_{k+1}\}$, где $a_i = b_i$ для $1 \le i \le r = |A \cap B|$. Если $r = k$ (т. е. $A \subseteq B$), то $A \cup \{b_{k+1}\} \in \mathsf{I}$ и условие $M2$ выполняется. Пусть $r < k$, и предположим, что $\langle a_1, \ldots, a_k \rangle$ является системой различных представителей для $\langle A_{i_1}, \ldots, A_{i_k} \rangle$, а $\langle b_1, \ldots, b_{k+1} \rangle$ является системой различных представителей для $\langle A_{j_1}, \ldots, A_{j_{k+1}} \rangle$ (индексы $i_1, \ldots, i_k$ попарно различны, как и $j_1, \ldots, j_{k+1}$). Рассмотрим следующее построение, которое либо находит такой элемент $b \in B \setminus A$, что $A \cup \{b\} \in \mathsf{I}$, либо заменяет последовательность $\langle A_{i_1}, \ldots, A_{i_k} \rangle$ некоторой новой последовательностью, для которой $\langle a_1, \ldots, a_k \rangle$ также является системой различных представителей.

Построение. Выберем такой индекс $p \le k+1$, что $j_p \notin \{i_1, \ldots, i_k\}$. Если можно так выбрать этот индекс, чтобы $r+1 \le p \le k+1$, то последовательность $\langle a_1, \ldots, a_k, b_p \rangle$ является

системой различных представителей для $\left\langle A_{i_1},\ ...,\ A_{i_k},\ A_{i_p}\right\rangle$, а из этого следует, что $A \cup \{b_p\} \in \mathrm{I}$. В противном случае, т.е. когда $1 \le p \le r$, имеем $a_p = b_p$ и последовательность $\langle a_1,\ ...,\ a_k\rangle$ является системой различных представителей для $\left\langle A_{i_1},\ ...,\ A_{i_{p-1}},\ A_{i_{p+1}},\ ...,\ A_{i_k}\right\rangle$.

Рассмотрим подробнее эту последовательность. Если последовательности $\langle i_1,\ ...,\ i_r\rangle$ и $\langle j_1,\ ...,\ j_r\rangle$ совпадают на $q < r$ позициях, то последовательности $\langle i_1,\ ...,\ i_{p-1},\ j_p,\ i_{p+1},\ ...,\ i_r\rangle$ и $\langle j_1,\ ...,\ j_r\rangle$ совпадают на $q+1$ позициях, так как $j_p \ne i_p$. Повторяя паше построение, мы либо находим требуемую частичную трансверсаль $A \cup \{b\}$, $b \in B \setminus A$ либо приходим к ситуации, когда $\langle i_1,\ ...,\ i_r\rangle = \langle j_1,\ ...,\ j_r\rangle$. Однако в последнем случае $j_p \notin \{i_1,\ ...,\ i_k\}$ для некоторого p, $r+1 \le p \le k+1$ (ведь не может быть $\{j_{r+1},\ ...,\ j_{k+1}\} \subseteq \{i_{r+1},\ ...,\ i_k\}$. Выполнение нашего построения еще раз приводит к нахождению такого элемента $b_p \in B \setminus A$, ÷òî $A \cup \{b_p\} \in \mathrm{I}$. ■

Матроид $M(\mathrm{A})$ называется *матроидом трансверсалей* семейства A .

Опишем теперь жадный алгоритм для матроидов трансверсалей. Этот алгоритм является модификацией метода нахождения систем различных представителей (а точнее, наибольшего паросочетания в двудольном графе), основанного на методике чередующихся цепей (Пусть $\langle A_1,\ ...,\ A_n\rangle$ – произвольная последовательность множеств (необязательно непересекающихся и необязательно различных). *Системой различных представителей для* $\langle A_1,\ ...,\ A_n\rangle$ будем называть такую произвольную последовательность $\langle a_1,\ ...,\ a_n\rangle$, что $a_i \in A_i$, $1 \le i \le n$, и $a_i \ne a_j$ для $i \ne j$. Мы будем говорить, что в такой системе различных представителей элемент a_i *представляет* множество A_i). Семейство $(A_1,\ ...,\ A_n)$ подмножеств множества $E = \{e_1,\ ...,\ e_m\}$ мы изображаем двудольным графом H с вершинами $u_1,\ ...,\ u_m,\ v_1,\ ...,\ v_n$ и рёбрами вида $\{u_i,\ v_j\}$, где $e_i \in A_j$. Каждой системе различных представителей $\langle e_{i_1},\ ...,\ e_{i_k}\rangle$ последовательности $\langle A_{j_1},\ ...,\ A_{j_k}\rangle$ (индексы $j_1,\ ...,\ j_k$ попарно различны) соответствует независимое множество рёбер $M = \{\{u_{i_1},\ v_{j_1}\},\ ...,\ \{u_{i_k},\ v_{j_k}\}\}$ в графе H .

Алгоритм 3.3. (*Жадный алгоритм для матроида трансверсалей*).

Данные: Семейство $\mathrm{A} = (A_1,\ ...,\ A_n)$ подмножеств множества $E = \{e_1,\ ...,\ e_m\}$,

представленное с помощью двудольного графа G. Элементы $e_1, \ldots, e_m$ заиндексированы в порядке невозрастания весов (веса неотрицательные).

Результаты: частичная трансверсаль S с наибольшим весом для семейства A (и паросочетание M графа H, определяющее функцию $\varphi : S \to \{1, \ldots, n\}$, о которой идет речь в определении частичной трансверсали).

```
1 begin
2   S:=∅; M = ∅;
3   for i:= 1 to m do
4     if существует в G чередующаяся цепь P относительно M с началом в e_i then
5       begin S:=S∪{e_i}, M:=P⊗M
6        end
7 end
```

Очевидно, что сложность этого алгоритма зависит от метода поиска чередующейся цепи P (строки 4 и 5). Если мы используем метод поиска в глубину, то число шагов, необходимое для отыскания цепи, будет порядка числа рёбер в графе G, т.е. $O\left(\sum_{i=1}^{n} |A_i|\right)$. Тогда общая сложность **алгоритма 3.3** будет $O\left(m\sum_{i=1}^{n} |A_i|\right)$.

Приведём теперь две задачи из практики, которые можно решать с помощью этого алгоритма.

Предположим, что некий предприниматель принял на работу n человек, причём i-й сотрудник имеет квалификацию, позволяющую выполнять любую работу из некоторого множества A_i.

Обозначим $\bigcup_{i=1}^{n} A_i = E = \{e_1, \ldots, e_m\}$ и предположим, что прибыль предпринимателя от выполнения работы e_j составляет $w(e_j)$ и что каждый из принятых на работу может выполнять не более одной работы. Нужно найти такое соответствие работ и. людей, могущих их выполнять, чтобы прибыль была наибольшей. На языке матроидов задача заключается в нахождении базиса с наибольшим весом матроида трансверсалей $M(\mathsf{A})$, где $\mathsf{A} = (A_1, \ldots, A_n)$. Точнее, нужно не только найти частичную трансверсаль S с наибольшим весом, но также и взаимно однозначную функцию $\varphi : S \to \{1, \ldots, n\}$, для которой $e \in A_{\varphi(e)}$, $e \in S$. Именно это и делает **алгоритм 3.3**.

Стоит отметить здесь, что, так же как и во всех жадных алгоритмах, представленных в данной главе, база с наименьшим (наибольшим) весом зависит только от упорядочения

элементов $e_1, \ldots, e_m$ по неубыванию весов. Обозначим через $e_i \le e_j$ тот факт, что $w(e_i) \le w(e_j)$ («работа e_j оплачивается не меньше, чем работа e_i»). Мы знаем, что множество работ $S = \{a_1, \ldots, a_k\}$ (где $a_1 \ge \ldots \ge a_k$) является оптимальным, а, следовательно, для каждого другого множества работ $T = \{b_1, \ldots, b_l\}$ (где $b_1 \ge \ldots \ge b_l$), которое может быть выполнено нанятыми на работу, имеем $l \le k$ и $b_i \le a_i$ для $1 \le i \le l$.

Второй пример касается нахождения оптимальной очерёдности выполнения работ. Предположим, что дано *n* работ $e_1, \ldots, e_m$, причём выполнение каждой из них требует одного и того же количества времени, скажем одного часа. Работы мы выполняем по очереди, не прерывая ни одной начатой работы, причём в каждый момент времени может выполняться не более одной работы. Для каждого i, $1 \le i \le m$, дан также *срок* d_i выполнения работы e_i (отсчитываемый от нулевого момента) и *штраф* $w(e_i)$, который выплачивается за невыполнение задания в срок (предполагаем, что штраф не зависит от того, на сколько часов был превышен срок). Нужно найти очерёдность выполнения работ, минимизирующую сумму штрафов. Эту же задачу мы можем сформулировать как нахождение выполнимого в срок множества работ с наибольшей суммой штрафов (тут мы максимизируем сумму штрафов, «которых избежали»). При этом нужно найти также очерёдность выполнения работ из этого оптимального множества. Связь этой задачи с матроидами трансверсалей следующая. Определим $E = \{e_1, \ldots, e_m\}$, $n = \max\limits_{1 \le i \le m} d_i$ è $A_i = \{e_j \in E : d_j \ge i\}$ для $1 \le i \le n$. При таких обозначениях произвольная частичная трансверсаль S семейства $\mathsf{A} = (A_1, \ldots, A_n)$ определяет выполнимое в срок множество работ, причём взаимно однозначная функция $\varphi : S \to \{1, \ldots, n\}$, такая что $e \in A_{\varphi(e)}$, $e \in S$, определяет для каждой работы $e \in S$ время $\varphi(e)$, в которое эта работа должна быть выполнена. Следовательно, оптимальное решение S является базисом с наибольшим весом матроида трансверсалей $M(\mathsf{A})$ и его можно найти, используя **алгоритм 3.3**.

Вопросы для самопроверки

1. Какие алгоритмы называются жадными ?
2. Дайте определение понятию матроид ?
3. Дайте определение понятию матроиду разбиений ?
4. Дайте определение понятию матроиду матрицы *A* ?

5. Дайте определение понятию частичная трансверсаль ?

6. Что является матроидом трансверсалей ?

7. Приведите примеры решения практических задач с помощью жадных алгоритмов для матричного матроида и матроида трансверсалей.

Задание на лабораторную работу

1. По заданным **алгоритмам 3.2** и **3.3** написать и отладить программы вычисления жадных алгоритмов для матричного матроида и матроида трансверсалей.

2. Продемонстрировать работающую программу преподавателю и получить отметку о ее выполнении.

3. Провести анализ полученного алгоритма.

4. Оформить отчёт о проделанной работе. Он должен включать в себя:

– цель работы;

– ответы на вопросы для самопроверки;

– схему обрабатывающего алгоритма и описание его работы;

– распечатки экранных форм, полученных в результате работы программы.

СПИСОК ЛИТЕРАТУРЫ

1. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. – М.: Наука. Гл. ред. физ.-мат. лит., 1990. – 384 с.

2. Калужнин Л.А., Сущанский В.И. Преобразования и перестановки: Пер. с укр. – 2-е изд., перераб. и доп. – М.: Наука, 1985. – 160 с.

3. Кофман А. Введение в прикладную комбинаторику. М.: Наука, 1975.

4. Сачков В.Н. Введение в комбинаторные методы дискретной математики. – 2-е изд., испр. и доп. – М.: МЦНМО, 2004. – 424 с.

5. Уилсон Р. Введение в теорию графов. М.: Мир, 1977. – 208 с.